E S C U E L A P O L I T E C N I C A N A C I O N A L

SIMULADOR DEL SISTEMA MICROPROCESADOR M6800 USANDO COMO BASE EL CROSS-ASSEMBLER DEL MISMO.

TESIS PREVIA A LA OBTENCION DEL TITULO DE INGENIERO EN ELECTRONICA Y TELECOMUNICACIONES.

EDISON ZUÑIGA HIDALGO

Quito Agosto de 1984

C E R T I F I C A C I O N

Certifico que el presente trabajo ha sido realizado por Edison Zúñiga bajo mi dirección.

Ing. Edgar Torres.

A G R A D E C I M I E N T O

Al Señor Jesucristo, quien es la motivación de mi vida y quien ha guiado la realización de este trabajo.

A mis padres, quienes se han sacrificado, en todo momento para poder desarrollar y culminar esta carrera estudiantil.

A mis hermanos, cuya ayuda ha sido inmejorable.

A los profesores de la Escuela Politécnica Nacional, cuyos conocimientos han servido para mi formación científica.

De manera especial un agradecimiento, al Señor Ingeniero Edgar Torres, Director de este trabajo, cuyos conocimientos intelectuales y humanos, han constituido la base para la realización cabal de este trabajo.

Al Departamento de Procesamiento Automático de Datos del IETEL Región 1, cuyo contingente humano y su infraestructura, han permitido la implementación del Simulador realizado.

P R O L O G O

El estudio a realizarse corresponde a la Simulación del Microprocesador M6800 usando como base el Cross-Assembler del mismo. Se hace necesario una simulación, puesto que , se requiere tener una comprobación de programas para el M6800, antes de ser aplicados a un sistema físico.

Este trabajo se origina, en los requerimientos, del Departamento de Procesamiento Automático de Datos del I. E.T.E.L R-1 ; de completar una obra previamente realizada que es el Cross-Assembler del microprocesador M6800 (el Cross-Assembler traduce los programas para el M6800, que están en lenguaje nmónico a lenguaje de máquina);pues el Cross-Assembler conjuntamente con el simulador, representan herramientas excelentes para el diseño y desarrollo de sistemas basados en microprocesadores, en este caso, el M6800.

El Sistema Simulador del M6800 a partir del código objeto generado por el Cross-Assembler, permitirá realizar corridas, pruebas simuladas, depuración de progra - mas simulados.

El lector deberá tener un conocimiento previo del microprocesador M6800 y si desea revisar los programas a implementarse, es necesario, el conocimiento del Fortran.

En el capítulo 1 se realizará un estudio de Simulación y de los Lenguajes de Alto Nivel, así como también se hace una justificación de la utilización del Fortran.

En el capítulo 2 se trata acerca de los Traductores e Intérpretes y las máquinas de estado finito como instrumentos para programación.

En el capítulo 3 se plantea el simulador, con los comandos que dispondrá y los errores que podría detectar.

La implementación del Simulador corresponde al capítulo 4; aquí se indica al simulador en diagramas de bloques y se presenta la documentación total de los programas y subrutinas realizados.

Se han diseñado varios ejemplos con el Simulador, los cuales son presentados en el capítulo 5; estos muestran en forma amplia el funcionamiento del simulador, así como su manejo.

Por último se muestra los listados de los programas realizados, y un manual de usuario, que es algo indispensable para el manejo del Simulador.

Quito, Agosto de 1984

I N D I C E

CAPITULO 1

1 INTRODUCCION Y CONCEPTOS PREVIOS..... 1

1-1 Introducción general y alcances..... 1

1-2 Simulación .-Conceptos..... 6

1-2-1 Sistema..... 6

1-2-1-1 Tipos de Sistemas..... 8

1-2-1-2 Subsistemas..... 11

1-2-1 Modelos..... 11

1-2-2-1 Clasificación de los modelos 12

1-2-2-2 Principios usados en modelación..... 13

1-3 Lenguajes de Alto Nivel.-Estudio y Análisis..... 14

1-3-1 Definición..... 14

1-3-2 Características..... 15

1-3-3 Elementos de los lenguajes de Alto Nivel..... 16

1-3-4 Ventajas y desventajas de los lenguajes de Alto Nivel..... 17

1-4 Comparación entre lenguaje Fortran y Lenguajes de Simulación..... 17

1-5 Justificación del uso de Fortran para la simulación del microprocesador M6800,..... 19

CAPITULO 2

2 SIMULACION DE ORDENADORES.-CONCEPTOS BASICOS Y CONSIDERACIONES PARA SU IMPLEMENTACION..... 21

2-1 Conceptos generales de Traductores e Interprétes..... 22

2-2 Máquinas de estado finito.-Justificación y ventajas de su utilización..... 25

2-2-1 Conceptos Preliminares..... 26

2-2-2 Máquinas de estado finito..... determinísticas y no determinísticas 30

2-2-3 Ventajas y desventajas para utilizar máquinas de estado finito..... 33

2-3 Matriz de transición..... 33

2-4 Implementación en tiempo real para optimización..... 35

CAPITULO 3

3 PLANTEAMIENTO DEL SIMULADOR DEL MICROPROCESADOR M6800..... 39

3-1 Consideraciones de la configuración GA-DM250:Memoria,Organización en disco,periféricos,sistema conversacional,transferencia del código Objeto proveniente del Cross-Assembler del M6800..... 40

3-1-1 Sistema GA-DM250..... 40

3-1-2 Código Objeto del Cross Assembler del M6800..... 41

3-2 Planteamiento Esquemático del Simulador.... 43

3-2-1 Esquema resumido del Simulador.. 43

3-2-2 Consideraciones de la aplicación del Código Objeto ensamblado y simulado a sistemas físicos... 45

3-2-3 Desarrollo del esquema del Simulador..... 46

3-3 Descripción del funcionamiento del Simulador..... 48

3-3-1 Submodelo de la Memoria del M6800..... 48

3-3-2 Submodelo del Decodificador de Instrucciones..... 49
3-3-3 Submodelo de los registros del M6800..... 50
3-3-4 Intérprete de instrucción direccionada.-Control de programa y Unidad Aritmética y Lógica simuladas..... 51

3-4 Comandos disponibles..... 52

3-4-1 Comandos para control de formatos e impresión de salidas..... 52
3-4-2 Comandos para poner valores.... 54
3-4-3 Visualización de valores..... 55
3-4-4 Comandos para ejecución de programa..... 56
3-4-5 Comandos de instrucción simulada 56
3-4-6 Macrocomandos..... 57
3-4-7 Comandos a tabla de macrocomandos 60

3-5 Errores que detectará el Simulador..... 60
3-6 Resultados que se obtendrán..... 64

CAPITULO 4

4 IMPLEMENTACION DEL SIMULADOR DEL MICROPROCESADOR M6800..... 67

4-1 Implementación del Simulador en Fortran. Archivos realizados..... 68

4-1-1 Definición del Sistema..... 68

4-1-2 Consideraciones de la implementación en Fortran..... 69

4-1-3 Archivos implementados..... 71

4-2 Diagramas de Bloques del Sistema..... 85

4-2-1 Objetivos de los diagramas.... 85

4-2-2 Simbología de los diagramas de bloques..... 86

4-2-3 Diagrama de bloques del Sistema Simulador del M6800..... 87

4-3 Descripción de programas y subrutinas.-Parámetros utilizados..... 93

4-3-1 Programas implementados..... 93

4-3-2 Subrutinas llamadas por programas anteriores..... 105

4-3-3 Subprogramas llamados por las subrutinas del punto 4-3-2.... 123

4-3-4 Subrutinas llamadas en los subprogramas de 4-3-3..... 151

4-4 Diagramas de Flujo de Programas y Subrutinas 178

4-4-1 Simbología para los diagramas de flujo..... 178

4-4-2 Diagramas de Flujo..... 181

CAPITULO 5

VARIOS EJEMPLOS DE APLICACION USANDO EL SIMULADOR IMPLEMENTADO.RESULTADOS Y EVALUACION DE LOS MISMOS 289

CAPITULO 6

CONCLUSIONES Y RECOMENDACIONES..... 322

APENDICES

APENDICE A :LISTADOS DE PROGRAMAS.....A1-A56

APENDICE B :TABLAS.-ARCHIVOS

APENDICE C :MANUAL DE USUARIO

BIBLIOGRAFIA

C A P I T U L O 1

INTRODUCCION Y CONCEPTOS PREVIOS

1.1 INTRODUCCION GENERAL Y ALCANCES

Se ha reunido en este estudio, por un lado una técnica que está siendo aplicada en muchas áreas de investigación, que es la Simulación y por otro el gran campo de los microprocesadores, con el M6800 como uno de ellos.

Las disciplinas científicas que utilizan la Simulación son diversas, es así, que se encuentra su aplicación en Economía, en Ingeniería, en Medicina, etc.

Por coincidencia también los microprocesadores cubren un amplio margen de aplicaciones, desde los juegos electrónicos hasta los más sofisticados controles en industrias, fábricas, etc.

El objetivo de este trabajo es realizar la Simulación del microprocesador M6800, considerando además, de los registros, del decodificador de instrucciones, de la Unidad Aritmética y Lógica; la memoria que el microprocesador puede accesar. Esta Simulación tendrá lugar a partir del Cross Assembler ya realizado en una tesis anterior(Tesis indicada en la Bibliografía).

La Simulación toma en cuenta modelos de Sistemas. El Microprocesador a Simular, es un Sistema Digital altamente integrado que contiene en un solo chip las siguientes unidades: Registro de Instrucciones y el Decodificador de Instrucciones, el contador de programa, un registro para el "stack-pointer", registros acumuladores, registros índices, la Unidad Aritmética y Lógica y un registro de condiciones. En la Simulación a realizarse se harán submodelos de las unidades enumeradas anteriormente, ya sea que se las tome por grupos o en forma individual. También se hará un submodelo de la memoria que se puede direccionar.

Se debe mencionar aquí, que las casas constructoras de microprocesadores(en este caso la Motorola) realizan estos programas de Simulación e indican su manejo(para referencia revisar el manual de Motorola M6800). Para la Simulación que se realizará se considera como guía el manual indicado anteriormente, pero se ha hecho cambios significativos; es así que directamente el Manual del Simulador del M6800 presentado por la Motorola no podría ser explicado en el Simulador a implementarse.

También se debe indicar que se han realizado estudios de Simulación de pequeños sistemas microprocesadores teóricos, con fines de explicación del funcionamiento de los mismos, tal caso se indica en el libro de Sherman(1970). Además como ejemplos de construcción de intérpretes o simuladores se han hecho pequeños trabajos como se muestra en el libro de Gries(1972). Estos datos serían como antece-

dentes históricos del trabajo que se desarrollará.

Para la conformación del Simulador del microprocesador M6800, es necesario tener conocimientos básicos como son: Las características del microprocesador M6800, su programación, la forma como funciona de acuerdo a las instrucciones que posee, especificaciones que presenta, etc.

Otros conocimientos que es indispensable para la programación del Simulador, es el lenguaje Fortran, sus características, los distintos tipos de sentencias que presenta; la formación de archivos mediante este lenguaje, etc. Algunas consideraciones de este lenguaje serán analizadas previamente.

Los conceptos de Simulación serán también necesarios conocer, entre estos se incluirán: el concepto de Sistema, de modelo, etc; todos estos conocimientos serán básicos para realizar la Simulación del M6800.

También se requiere saber acerca de los Traductores e Interpretes, lo que estos representan y ciertas nociones sobre como se los construye.

Para la implementación de Simulador, dentro del computador que se utilizará; es imprescindible conocer el manejo de este computador, su configuración, la etapa para la realización de un programa como son: Edición, compilación, depuración, ejecución, ingreso de datos y obtención de resultados.

Se había indicado que la Simulación tomará en cuenta el microprocesador con sus registros, decodificador de instrucciones, Unidad Aritmética y Lógica, memoria a direccionarse .No se simulará los circuitos exteriores como son: circuitos de reloj y de control de datos o direcciones, circuitos para operaciones de inicialización e interrupciones. Unicamente se harán interrupciones simuladas luego de una instrucción WAI, ingresadas como datos de comandos de interrupción al Simulador. Por lo tanto la simulación de subrutinas para el control de aparatos de periferia,será de manera restringida; pues considerando que los equipos de interface serán únicamente localidades de memoria; en estas pueden ser gravados datos para ser probados pero no podrán ser cambiados a lo largo de una ejecución,como sucede en la realidad con los interfaces de la periferia.

El método seguido es un método científico con sus pasos: Observación del sistema físico en este caso el microprocesador M6800; formulación de una hipótesis(en este estudio corresponde al modelo realizado); investigación del comportamiento del sistema(en un programa en computador construído para el modelo), experimentos de validación (diseño de estos; ejecución de corridas de simulación y análisis de resultados).

El plan de trabajo seguido es el siguiente: primero se ha realizado un estudio de: simulación, lenguajes de alto nivel, traductores e interpretes, máquinas de estado finito, matriz de transición; y el conocimiento del computador que se utilizará,en un tiempo de 4 meses. La imple

mentación de programas, documentación de programas, diseño de experimentos, corridas de simulación, etc., ha tomado un tiempo de 8 meses.

Los resultados obtenidos son satisfactorios, pues cumplen los objetivos de los programas simulados como ejemplos.

1-2 SIMULACION.-CONCEPTOS

Para el estudio de Simulación se expondrá ciertos criterios sobre lo que representa un Sistema,los elementos del mismo,la clasificación que se hace de Sistemas;además se dará a conocer lo que significa un modelo,como se lo realiza,las clases de modelos que existen,etc.

Debido a la necesidad de estudiar ciertos fenómenos por un método más económico que al hacerlo en la realidad y en vista que,en un estudio de estos,sería aconsejable realizarlo tomando en consideración únicamente las características más fundamentales;se requiere de una Simulación.

Se conoce por Simulación a la técnica numérica que realiza experimentos en un computador,sobre modelos que representan situaciones físicas de un Sistema;los experimentos toman en cuenta el tiempo en que hay cambios en el modelo matemático del Sistema.

Se verá a continuación lo que se entiende por Sistema ,siendo este término utilizado en muchas áreas de conocimiento,teniendose por ejemplo : un Sistema de Transmisión, un Sistema de Tráfico Vial,el Sistema Planetario,etc.

1-2-1 SISTEMA

Del estudio de algunos ejemplos de Sistemas,se

desprende que un Sistema es un conjunto de objetos que interactúan entre sí;los objetos de un Sistema que son de interés para un determinado estudio se los conoce como entidades del Sistema;se reconoce además como otro elemento de los Sistemas a las propiedades que tienen las entidades,a estas se les llama atributos y debido a que hay cambios en el Sistema se busca los procesos que causan estos cambios y se les denomina como actividades del Sistema.

Para aclarar los conceptos anteriores se reconocerá los elementos de Sistemas en los siguientes ejemplos:

Ejemplo 1-1

En un Sistema de Transmisión de una señal de radiofrecuencia,los objetos de interés o entidades,entre otros serían:el transmisor,el receptor,el canal de transmisión,etc; con sus propiedades o atributos como:la frecuencia,la potencia de transmisión,la capacidad del canal de transmisión,el ancho de banda de la señal,etc;habiendo cambios en el Sistema causados por las actividades como:la modulación de la señal,la interferencia de la señal,etc.

Ejemplo 1-2

En un Sistema de Tráfico Vial,los elementos que se reconocen son:los vehículos,la carretera,las leyes de tránsito,etc;los atributos de este Sistema representarían:las dimensiones de los vehículos,las velocidades que desarro-

llan,tipo de carretera,reacciones de los conductores de acuerdo a la carretera,etc;y teniendo como algunas actividades:obstaculos en la carretera,comportamiento delos otros vehículos,etc.

Cuando se describe un Sistema en una circunstancia de tiempo y en un punto determinado se dirá que este es un estado del Sistema,en el ejemplo anterior del tráfico vial puede ser que interese describir la circulación vehicular en una carretera determinada y a una hora específica,siendo este,el estado del tráfico.

1-2-1-1 TIPOS DE SISTEMAS.-

Con el propósito de clasificar a los Sistemas se ha tomado distintos puntos de vista,por ejemplo se ha considerado los tipos de actividades que existen en un Sistema,también se ha clasificado por el efecto que causan las actividades en un Sistema.

Un Sistema es afectado en la mayoría de los casos por cambios ocurridos fuera del mismo,también conocidos como cambios del medio ambiente del Sistema,llegando a este como actividades exógenas; siendo por lo tanto endógenas, las actividades que ocurren dentro del Sistema;de acuerdo a lo anterior ,un Sistema si no tiene actividades exógenas es um Sistema Cerrado,mientras que un Sistema se llama Abierto en caso contrario.

En el ejemplo anterior de un Sistema de Transmisión se visualiza como actividades exógenas:el ruido que afecta a la señal,o desde otro punto de vista:las recomendaciones, los protocolos y normas que se hagan para una correcta transmisión;clasificandose por esto como Sistema Abierto.Se debe considerar que aquí se ha supuesto como medio ambiente del Sistema de Transmisión, a los factores que se han tomado en cuenta para decidir sobre las recomendaciones para una transmisión;pero para otro tipo de estudio puede ser que se aborde también estos factores,en este caso estos factores serán actividades en el Sistema; de aquí que es conveniente limitar el estudio de un Sistema y ver cual es su medio ambiente.

Cuando por efecto de una actividad se produzcan resultados completamente dependientes de esta,será una actividad determinística;pero si los resultados varían en algunos posibles valores,se tendrá una actividad estocástica; esta es otra clasificación de actividades.

Con otro enfoque ,los Sistemas pueden ser:Contínuos, cuando los cambios que se produzcan en un Sistema sean predominantemente suaves y es Discreto cuando los cambios sean discontinuos;para visualizar lo anterior vale el siguiente ejemplo :En un Sistema de muestreo de datos,el Sistema es intrínsecamente contínuo pero la información que proporciona es unicamente disponible en puntos discretos del tiempo.

En un Sistema Discreto se utilizará un set de nú-

meros ,cada uno representando un aspecto del estado del Sistema ;a estos números se les suele denominar como descriptores del Sistema,pudiendo ser de dos categorías:descriptores con significado físico y descriptores de condiciones.

En el ejemplo del tráfico vial analizado anteriormente,se considera la forma discreta de llegada de vehículos a un cruce,pudiendose describir el Sistema con números que indiquen los vehículos que llegan,número de carriles de carretera,etc;y números que den cierta condición,como la disponibilidad de cruzar una vía.

Para ampliar el conocimiento anterior se reconocerá los descriptores en el siguiente ejemplo:

Ejemplo 1-3

En un Sistema de Transmisión de Datos,tendrán significado físico ,los descriptores del Sistema que representan el mensaje enviado;mientras que las instrucciones como:'enviar señal','señal tiene error',etc;tendrán descriptores que significan una condición.

Los descriptores del estado del Sistema pueden cambiar debido a muchas circunstancias,a estas se les llama eventos.Así en el ejemplo anterior,un evento sería la corrección de error en un mensaje.

Cuando se realiza el estudio de un Sistema relativa-

mente grande se da la posibilidad de enfocarlo por partes denominadas Subsistemas.Se verá a continuación lo que estos representan.

1-2-1-2 SUBSISTEMAS

Se acostumbra a describir un Sistema como intercambio de subsistemas,organizándoles en bloques e identificando entradas y salidas a cada bloque.

Cada Subsistema es un bloque y hay relaciones entre entradas y salidas;estas relaciones,deben ser suficientes para determinar salidas en función de entradas y considerar cada bloque como una caja negra.

Las salidas de subsistemas pueden ser entradas a otros,siendo estas:variables endógenas al Sistema.

1-2-2 MODELOS

El estudio de los fenómenos en la realidad pueden presentar algunas dificultades,así por ejemplo en un Sistema Económico,donde se tienen como objetos de estudio:la oferta,la demanda,los precios,entre los principales;sería muy costoso el ir variando las ofertas para ver como resultarían los cambios en los precios;por lo cual hay la necesidad de realizar una representación adecuada del Sistema;esta representación que debe ser una aproximación lo más cercanamente posible a la realidad y una simplificación

del Sistema ,se conoce como Modelo del Sistema,que es un cuerpo de información recolectada acerca del Sistema con el propósito de estudio del mismo.

También se justifica la utilización de modelos cuando no se alcanza a visualizar los fenómenos,como es por ejemplo un modelo de bandas que se hace para estudiar la conducción eléctrica en un conductor.

La tarea de establecer un modelo comprende:

-Estructurar el modelo,identificando:entidades,atributos y actividades del Sistema.

-Asignar los datos o valores que los atributos pueden tener y la relación entre las actividades.

1-2-2-1 CLASIFICACION DE LOS MODELOS

Se acostumbra a clasificar los modelos así:

-MODELOS FISICOS.-Se basan en analogías entre Sistemas;ejemplo sistemas hidraúlicos con sistemas eléctricos;en estos casos,los atributos pueden ser señales medibles como el voltaje y las actividades son las leyes físicas.

-MODELOS MATEMATICOS.-Usan notación simbólica y ecuaciones matemáticas que relacionan las variables.Por ejemplo el modelo matemático del Sistema Planetario son las ecuaciones de Kepler.

-MODELOS ESTATICOS.-Muestran los valores de los atributos, cuando el Sistema está en equilíbrio,no toman en cuenta la variable tiempo;un ejemplo de un modelo físico-estático es la representación de los átomos por esferas.

-MODELOS DINAMICOS.-Tratan de las interacciones que varían con el tiempo.

Para los modelos matemáticos,hay los métodos:Analíticos y Numéricos.

Analítico.-Utiliza deducciones de la teoría matemática y busca un modelo que resuelva,como por ejemplo: una ecuación diferencial.

Numérico.- Incluye procesos de computación para resolver ecuaciones.

La Simulación se considera una técnica numérica de computación,utilizada para modelos matemáticos-dinámicos.

1-2-2-2 PRINCIPIOS USADOS EN MODELACION

Se consideran ciertas reglas para construir modelos matemáticos y juzgar la información incluída en un modelo.

a)Construcción de diagramas de bloques.-Para simplificar las especificaciones de la interacción dentro del Sistema,cada bloque describe una parte de este.

b)Se incluyen aspectos propios al objetivo de estudio.

c)Se debe tomar en cuenta la acuciosidad de la información guardada.

d)Un número de entidades individuales pueden ser agrupadas juntas en largas entidades.

1-3 LENGUAJES DE ALTO NIVEL.-ESTUDIO Y ANALISIS

Se ha incluído en las siguientes secciones del Capítulo un estudio de lenguajes de alto nivel,lenguajes de Simulación y el lenguaje Fortran que es el que utilizaremos en la Simulación del microprocesador M6800.

Al crearse los lenguajes de alto nivel,se obtuvo muchas ventajas sobre el lenguaje de máquina,y se incluyeron en el lenguaje de alto nivel,los elementos necesarios para la formulación de sentencias de forma semejante a los algoritmos que realizan los programas;todo esto será motivo de análisis en este punto.Se empezará definiendo a los lenguajes de alto nivel y describiendo las características de estos,de la siguiente manera:

1-3-1 DEFINICION

Lenguaje de Alto Nivel es aquel,en que cada instrucción o sentencia corresponde a varias instrucciones de có-

digo de máquina;se los llama también lenguajes algorítmicos,porque cuando se escriben algoritmos,sus pasos se pueden traducir directamente a sentencias.

Las sentencias de los lenguajes de alto nivel se han desarrollado para parecerse mucho a fórmulas a ser evaluadas,siendo sentencias aritméticas o de asignación,habiendo también sentencias para entrada y salida,movimiento de datos y operaciones de decisión.

Estos lenguajes permiten a los usuarios escribir en una notación con la que están familiarizados,por ejemplo Fortran en una notación matemática.

Mediante lenguajes de alto nivel,se describen programas que dan la lógica de los procedimientos,antes que el trabajo de la computadora.

A continuación se expondrá las principales características de los lenguajes,de acuerdo a como estos se presentan y como se desarrollaron.

1-3-2 CARACTERISTICAS.-

-Tienen aparencia natural para una clase particular de problemas,usando notación matemática.

-Son formales,es decir,obedecen a reglas sintácticas(reglas de la estructura de las sentencias).

Dentro de un programa se presenta lo que se denomina control de flujo,que se refiere a que generalmente el control del programa pasa de una sentencia a la siguiente,a no ser que haya una transferencia de control.

1-3-3 ELEMENTOS DE LOS LENGUAJES DE ALTO NIVEL

Se ha incluído aquí los siguientes:

-Constantes.-Cantidades que no cambian durante la ejecución, pudiendo ser:enteras,reales,alfanuméricas.

-Variables.-Son números que pueden tomar diferentes valores en el programa;las variables pueden ser:enteras, reales,lógicas.

-Funciones.-Son algoritmos desarrollados para el cálculo de algunas funciones matemáticas como son por ejemplo:sen(x),cos(x),etc;en el programa se proporciona el argumento x para el cual se desea saber el valor de la función.

-Operaciones y expresiones.-Cuatro operandos aritméticos,exponenciación,ciertas relaciones y operaciones lógicas.

Para dar un valor a una variable se utiliza sentencias de asignación,y para describir el tipo de datos se utiliza declaraciones.

1-3-4 VENTAJAS Y DESVENTAJAS DE LOS LENGUAJES DE ALTO NIVEL

Considerando que existen muchas ventajas y ciertas desventajas de los lenguajes de alto nivel,se indican aquí las principales.

Se tiene facilidad y rapidez en la escritura de programas,pues hay mejor control de software;también facilidad en la depuración,menor costo de programación,y siendo los programas,independientes del trabajo interno de la computadora,no se necesita de su configuración.

Otra ventaja es la documentación propia de los programas,facilitandose el mantenimiento de los programas,ya que en sí son explícitos.

Entre las desventajas,podemos citar,que debido a que los programas son independientes de la máquina,no se puede aprovechar todas las posibilidades de manejo,o en su lugar los programas traducidos,a veces tienen códigos ineficientes;como otra desventaja se puede decir que se forman grandes programas aumentando el costo de compilación

1-4 COMPARACION ENTRE LENGUAJE FORTRAN Y LENGUAJE DE SIMULACION

El lenguaje Fortran es un lenguaje natural y formal, las sentencias son cercanas a las sentencias matemáticas que usaríamos para escribir los procedimientos,es el lenguaje más utilizado,de mayor disponibilidad y cuyo conocimiento está bastante difundido.La ventaja principal con respecto

a lenguajes de Simulación es la flexibilidad en:diseño y formulación de modelos,tipo y formato de reportes de salida,clases de experimentos de Simulación y también la interdependencia con el diagrama de flujo.

En Fortran el número de subrutinas es limitado básicamente por la capacidad de almacenamiento y se tiene flexibilidad de escribir cualquier subrutina que se necesite para programas particulares de Simulación;con lo que se puede construir un programa de Simulación a partir de un conjunto de subrutinas ya programadas.

Las instrucciones disponibles en Fortran son:fórmulas aritméticas,proposiciones de control,proposiciones de entrada y salida,proposiciones de especificación.Además se tiene cinco componentes que son:variables,constantes,subindices,expresiones y funciones.

Sobre los lenguajes de Simulación,se dirá que,los más conocidos son:GPSSII,SIMSCRIPT,SIMIAC,DYNAMO,etc;estos lenguajes simplifican la labor de programación,ya que se aplican directamente al sistema a simularse,tienen estructura generalizada para el diseño,presentan una forma rápida para introducir cambios y una forma flexible para obtener reportes.

Con lenguajes de Simulación puede haber reducción en tiempo de programación,pero se incrementa el tiempo de computo y costo.Estos lenguajes se adaptan mejor a problemas de planificación y a sistemas económicos.

1-5 JUSTIFICACION DEL USO DE FORTRAN PARA LA SIMULACION DEL MICROPROCESADOR M6800

Siendo el compilador Fortran de carácter universal y por disponer de este compilador en el computador a usarse,se ha optado por realizar la Simulación del Microprocesador M6800 en lenguaje Fortran y además por que en el procedimiento de Simulación se dan los siguientes elementos básicos:

-Valores Iniciales.-Por medio de Fortran,el programador encuentra facilmente el método de leer valores iniciales o generarlos.

-Generación de Datos.-En base a condiciones iniciales y valores de variables exógenas leídas o generadas,la computadora genera variables endógenas de acuerdo al Sistema;con Fortran la generación de variables exógenas depende del programador.En el sistema microprocesador M6800 las variables son leídas (estando estas,en un programa en lenguaje de máquina para este microprocesador);la computadora simuladora genera variables endógenas,que serían,los valores que van tomando los registros simulados o los valores en memoria simulada.

-Mecanismo para flujo de tiempo.-Hay métodos de programación con incrementos fijos y variables de tiempo,con Fortran hay libertad para escoger cualquiera de los dos métodos.

En el caso de la Simulación del microprocesador M6800

necesitamos incrementos variables de tiempo,ya que el registro de tiempo simulado,toma en cuenta la demora de ejecución de las instrucciones,siendo variable para cada instrucción.

-Fortran ofrece máximo de posibilidades de diseñar reportes de salida,sea gráfica o tabular;en el sistema a simular necesitamos una salida tabular de registros,áreas de memoria,etc.

Otro aspecto que se ha tomado en cuenta es que:no hay disponibilidad de compiladores de Simulación en la mayoría de computadores,mientras que el compilador Fortran es ampliamente utilizado.

C A P I T U L O 2

SIMULACION DE ORDENADORES.-CONCEPTOS BASICOS Y CONSIDERACIONES PARA SU IMPLEMENTACION

Luego de que se ha expuesto en el capítulo anterior ciertos criterios de Simulación;en este capítulo se realizará al inicio,una breve visualización sobre traductores para tener una mejor idea sobre el traductor Cross-Assembler que acoplaremos al Simulador;y en el mismo punto se definirá a las rutinas interpretativas o simuladoras,complementandose con un ejemplo de un interprete;es decir enfocandose ya la Simulación de ordenadores o procesadores que es el objetivo previsto;además debido a que el Simulador que se pretende realizar necesita de comandos de control para el ingreso de datos y otras operaciones,y como estos mecanismos de control, en sí,son un lenguaje para el manejo del Simulador,se requiere de instrumentos de programación para este lenguaje,que son las máquinas de estado finitas o autómatas de estado finitas;de estas se realizará un análisis y un estudio de su representación en el computador en la matriz conocida como Matriz de Transición.

Para terminar este capítulo se verá algunas consideraciones sobre la implementación de programas en tiempo real.

2-1 CONCEPTOS GENERALES DE TRADUCTORES E INTERPRETES

Como se dijo en el punto 1-3,se ha requerido de lenguajes de alto nivel,pues son los que tienen muchas ventajas en programación sobre los lenguajes de máquina;de aquí que hubo la necesidad de tener traductores de un lenguaje a otro.En este punto se tratará de estos traductores y también se considerará a los intérpretes.

Un traductor ,es un programa que traduce un programa fuente a un programa objeto equivalente.El programa fuente está escrito en un determinado lenguaje fuente,Fortran por ejemplo;el programa objeto se representa en un lenguaje objeto,que podría ser código de máquina.La ejecución de esta traducción se realiza en un tiempo de traducción.

Sí el lenguaje fuente es un lenguaje de alto nivel tal como Fortran,PL/I,Cobol,etc,y el lenguaje objeto es uno de bajo nivel tal como lenguaje Assembler o lenguaje de máquina,al traductor se lo conoce como compilador.

Sí el programa fuente está en lenguaje ensamblador y es traducido a lenguaje de máquina ,al traductor se lo llama ensamblador.

En el caso de que un programa escrito para un ordenador sea ejecutado por otro,la rutina que ejecuta,se conoce como rutina interpretativa o simuladora.

Es un intérprete para lenguaje fuente,aquel que ejecuta programas escritos en lenguaje fuente.Esta ejecución puede efectuarse en dos pasos:

1-Se traduce un programa fuente escrito en algún lenguaje fuente (Fortran por ejemplo) a un formato interno.

2-Se ejecuta(interpreta o simula)el programa en esta forma interna.

Por lo general,se conoce como interpretación la segunda parte;ya que interpreta la forma interna,ocurriendo esta en un tiempo llamado tiempo de interpretación.

Debemos anotar que este tiempo de interpretación es un poco largo,siendo un programa intérprete mucho maś lento en ejcución que un programa equivalente en lenguaje de máquina;no debiendo emplearse en trabajos rutinarios,donde gran parte del tiempo del ordenador se emplea en ejecución.

Ejemplo 2-1

Como ejemplo se realizará aquí,la interpretación de un istema computador digital imaginario.Supongase que tenga 16 celdas de memoria,direccionadas :0000_2,------,1111_2, que disponga de un registro acumulador,un registro de direcciones,un registro de instrucciones y que tenga las instrucciones siguientes:

Instrucción	Descripción	Código
Load	Colocar un número en acumulador	0010
Store	Almacene un número del acumulador en memoria	0011
Add	Sume un número al acumulador	0100

En la simulación o interpretación de este pequeño sistema,se reconoce como entidades del sistema:los registros, memoria,instrucciones;pudiendo afirmarse que los atributos del sistema serían:la longitud de los registros(4 bits),número de registros,capacidad de la memoria(16 celdas),longitud de la palabra de memoria(4 bits);y como actividades del sistema serían:datos,resultados intermedios,proceso de los datos.

En este ejemplo ,el ambiente del sistema,que puede contener:las conexiones exteriores como bus de datos,bus de direcciones,etc;no tomaría parte en la Simulación.

A este Sistema se lo clasificaría como discreto por tener cambios discontinuos,es un sistema abierto por que es afectado por factores exógenos tales como:arribo de datos, programas en lenguaje de máquina que llegan a este,etc.

Los descriptores de este sistema serían los registros simulados,los cuales tienen significado físico;en este caso por no haber registro de condiciones,no hay descriptores de condición.

Para la Simulación de este sistema se formaría un modelo matemático,donde solo se tomaría en cuenta:los registros,memoria,instrucciones,sin incluir en el modelo,el ambiente del sistema,como son las conexiones exteriores que tendrían los registros.Este modelo matemático es dinámico ya que se consideraría el tiempo que va a tomar cada instrucción y se utilizaría un método numérico ,pues se haría el interprete en un computador simulador mediante las operaciones numéricas que nos permita usar este.

2-2 MAQUINAS DE ESTADO FINITO .-JUSTIFICACION Y VENTAJAS DE SU UTILIZACION

Se ha creído conveniente incluir este punto en razón de que se va a realizar un conjunto de comandos para manejo del Simulador(lenguaje para controlar el Simulador);para esta realización,se requiere de un reconocedor o máquina de estado finito,que pueda indicarnos si están bien conformados los grupos de caracteres ingresados por el usuario del Simulador.

El programar en base a estas máquinas abstractas llamadas máquinas de estado finito,dará ciertas ventajas y desventajas,las cuales serán brevemente analizadas al final de este punto;también,para desarrollar de una manera organizada el conocimiento de estos autómatas o máquinas,se hace una introducción de conceptos,los mismos que ayudan a definir a las máquinas de estado finitas y todo esto tiende a que se cumpla el objetivo de realizar un lenguaje de manejo del

Simulador.

2-2-1 CONCEPTOS PRELIMINARES

Sin querer profundizar mucho en algunos conceptos, se dará una idea de los principales tales como:Gramática, sintaxis y semántica,alfabeto,lenguaje y culminar con la definición formal de una gramática regular.

Considerese una expresión algebraica:

i+i

Puede decirse que esta expresión cumple ciertas reglas para su formación,considerandolas como parte de la sintaxis del Algebra;pues si aparecería así:

ii+

esta ya no sería una expresión algebraica.

También la expresión expuesta tiene un significado que es la suma de dos variables,este significado representa la semántica corriente del Algebra.

Hasta aquí se ha expresado que existen ciertas reglas sintácticas ;y cuando se tiene un conjunto finito y no vacío de reglas ,se dice que hay una Gramática G;las

partes constitutivas de las reglas se denominan unidades o clases sintácticas;las reglas están relacionadas por un signo '::=',que quiere decir :'puede contener';en las reglas aparece también el signo :'| ',que indica las alternativas que tiene la parte izquierda en ir a la derecha de una regla; además las unidades sintácticas se presentan entre paréntesis angulares.Todas las afirmaciones anteriores se ilustran en el siguiente ejemplo.

Ejemplo 2-2

Sea una gramática para expresiones algebraicas con un operando i y operadores:+ , - , . ,/ ;las reglas para esta gramática se podrían formar así:

<expresión> ::= <término> | <expresión + término>
<expresión>::= <expresión - término>
<término > ::= <factor>|<término . factor>
<término > ::= <término>/<factor>
<factor > ::= <expresión>|i

Aquí se ve,que de acuerdo a la nomenclatura dada, la primera regla,por ejemplo,muestra que una expresión puede contener un término ,o una expresión más un término.

LLamase alfabeto a un conjunto finito y no vacío de elementos o símbolos de una gramática;tira a una serie finita de símbolos de un alfabeto,así por ejemplo para un alfabeto:

$$A = \{a,b,c\}$$

las tiras de A podrían ser:

a , b , c , ab , aaca , ε

donde ε representa la tira nula.

Los símbolos se llaman no terminales cuando aparecen en la parte izquierda de una regla,por ejemplo <expresión> en la gramática del ejemplo 2-2;y se denomina símbolo distinguido z de la gramática G(z),al que debe aparecer por lo menos una vez en la parte izquierda de una regla.

Se conoce como lenguaje al subconjunto del conjunto de todas las tiras terminales(tiras compuestas por símbolos terminales),en el ejemplo 2-2 para el símbolo terminal i, el lenguaje podría contener :

i , i.i ,i+i.i ,i-i.i , i.i/i+i ,etc

A continuación se define formalmente una gramática regular.Para esto,sea el arreglo (V,T,P,Z) donde :V es un alfabeto,T contenido en V,es el alfabeto de símbolos terminales,P es un conjunto finito de reglas y Z es el símbolo distinguido.Se llama gramática regular si esta cumple con la o las reglas:

U ::= N

U ::= WN

En las reglas anteriores N está en T,y U y W están en V-T; siendo V-T el alfabeto de símbolos no terminales.

Para mayor información de lo que es la gramática regular,demos un ejemplo de esta:

Ejemplo 2-3

Consideremos una gramática G(entero) para generar el campo de los números enteros,el conjunto de reglas de esta gramática podría ser:

<entero> ::= dígito |<entero> dígito

<dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Identificando las partes de esta gramática con la nomenclatura dada en la definición anterior,se afirma qué:

V,el alfabeto de la gramática,será:

$$V = \left\{ <\text{entero}>, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \right\}$$

T,el alfabeto de símbolos terminales:

$$T = \left\{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \right\}$$

P,son las reglas indicadas al principio del ejemplo

Z,el símbolo distinguido es : <entero> en la gramática de los enteros G(entero)

Con todos estos conceptos preliminares,se pasa a analizar lo que se entiende por máquinas de estado finito.

2-2-2 MAQUINAS DE ESTADO FINITO,DETERMINISTICAS Y NO DETERMINISTICAS

Se las conoce como autómatas,máquinas o reconocedores de estado finito,y sirven para manipular modelos de máquinas.

No es un objeto real,sino un modelo matemático con propiedades y comportamientos específicos y que se pueden simular con un programa de computadora.

Un autómata de estado finito determinista (AF) es un arreglo de (K,VT,M,S,Z) donde:

K :alfabeto de estados
VT :alfabeto de entrada (caracteres de una tira)
M :función que pasa de un K y VT a otro K (sí M(Q,T)=R del estado Q con el caracter T pasamos a R)
S :estado inicial
Z :conjunto de estados finales

Es determinista cuando en cada paso,el siguiente caracter de entrada determina univocamente el próximo estado.

Para ver un ejemplo de un autómata finito se indica primeramente como construir un diagrama de estados.

Ejemplo 2-4

Para la gramática G(Z)con las reglas:

Z ::= U0|V1
U ::= Z1 | 1
V ::= Z0 | 0

Esta gramática produce un lenguaje de ceros y unos, en parejas de 01,10.

Cada símbolo no terminal se pone como un nudo o estado,se tiene también un estado inicial S con su correspondiente nudo;y para las reglas siguientes se ilustra con arcos las transiciones,así:

U :: 1 hay un arco de nombre 1 que va de S a U
U ::Z1 hay un arco 1 que va de Z a U

Entonces,si al diagrama de estados se le incluye un estado de'fallo' F,quedaría:

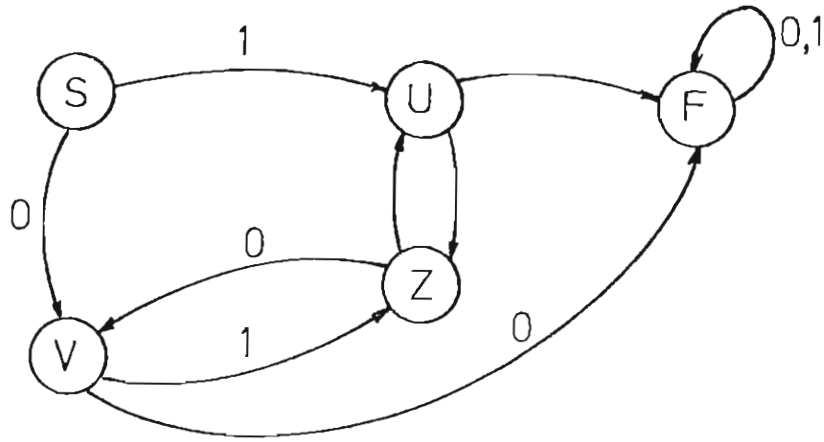


Fig.2-2-2-1

Considerando las reglas de la gramática G(z) y el diagrama de estados;el autómata de estado finito sería:

$$AF\left(\{S,Z,U,V,F\},\{0,1\},\ M\ ,\ S\ ,\{Z\}\right)$$

M(S,0)=V	M(S,1)=U
M(V,0)=F	M(V,1)=Z
M(U,0)=Z	M(U,1)=F
M(Z,0)=V	M(Z,1)=U
M(F,0)=F	M(F,1)=F

La máquina de estado finito no determinística,se diferencia de la determinística,en que,la función M da un conjunto de estados siguientes,en vez de un estado siguiente único;además puede tener varios estados iniciales o también varios estados finales y tiene reglas como:

V :: UT Y W ::= UT

Los autómatas finitos no determinísticos,pueden ser transformados a determinísticos,sustituyendo un conjunto de

estados por un estado equivalente;si por ejemplo hay elec - ción de 3 estados X,Y,Z , el autómata finito tendrá un estado único ; (X,Y,Z).

2-2-3 VENTAJAS Y DESVENTAJAS DE UTILIZAR MAQUINAS DE ESTADO FINITO.-

De acuerdo a la aplicación se utiliza o no máquinas de estado finito;ya que con el uso de estas aumenta la rapidez de la ejecución pero aumenta también el uso de memoria.

Generalmente se utiliza máquinas de estado finito cuando hay que manejar problemas recurrentes de compilación, o cuando se necesita tener flexibilidad de cambios en un reconocimiento de estados.

Al simular una máquina de estado finito se incrementa el almacenamiento de memoria ,pero se simplifica los problemas de manejo de memoria.

2-3 MATRIZ DE TRANSICION

Es la representación dentro de un computador u ordenador de un autómata de estado finito;así,si los estados son:S1,S2,S3,.....,Sm , y los caracteres de entrada son: T1,T2,.....Tn;se representan en una matriz de transición B de mxn.

Cada elemento de la matriz es B(i,j) y contiene un estado siguiente Sk cumpliendose que M(Si,Tj)=Sk;siendo M la función que de un estado Si con un caracter de entrada Tj pasa al estado Sk. Por pasarse de un estado a otro se llama matriz de transición.

Las columnas de la matriz son etiquetadas con símbolos o caracteres de entrada ,mientras que las filas representan los estados;las entradas generan nuevos estados,en una columna de entrada y una fila de estados.

La primera fila de la matriz contiene los estados iniciales,y a la derecha de las filas,formando la última columna se pone un indicativo de estados aceptores con un'1', y de estados no aceptores con un'0';esta columna se nombra con un signo '+'.

En el ejemplo anterior de máquinas de estado finito para la gramática que da la secuencia de parejas 01,10;la matriz de transición se expresaría así:

		Símbolos de entrada		
		0	1	+
	S	V	U	0
Estados	U	Z	F	0
	V	F	Z	0
	Z	V	U	1
	F	F	F	0

FIG. 2-3-1

En la matriz de la Fig.2-3-1 ,se observa que estando en el estado S e ingresando un caracter 0 se pasa al estado V; el estado aceptor es Z,y los demás estados no son aceptados.

La matriz de transición se empleará cuando por autómatas de estado finito ,se reconozcan grupos de caracteres válidos,en instrucciones que facilitarán el manejo del Simulador.

2-4 IMPLEMENTACION EN TIEMPO REAL PARA OPTIMIZACION

En esta sección se darán ciertos criterios ,que son útiles para la implementación de programas en el computador, que tienen que ver ,en el caso del Simulador a implementarse, con los mecanismos de control de este,que existirán en tiempo de ejecución,y también con la forma de optimizar programas para disminuir el tiempo de compilación y de ejecución de los mismos.

Se dice que un sistema está en tiempo real,cuando el proceso de entrada de datos al sistema,para obtener el resultado,se produce de forma virtual y simultanea con el proceso que genera los datos;es decir el ordenador debe ser capaz de recibir y transmitir datos suficientemente rápido y debe poder obtener inmediatamente información de la memoria.

Los sistemas en tiempo real,requieren :un equipo de comunicación de datos para alimentar estos desde un terminal,

dispositivos de almacenamiento de acceso directo,utilización de unidades centrales de proceso,en tiempo compartido;la finalidad de un sistema en tiempo real,es que el ordenador proporcione una imagen de los sucesos a medida que se producen, por tanto,un sistema de este tipo debe tener alguna forma de recuperación de información , como por ejemplo unidades de representación visual.

Para un diseño de un Simulador en tiempo real,se debe planear como va a ser la salida del mismo,es decir el tipo de estructura de datos y mecanismos de control que existirán en tiempo de ejecución,por ejemplo:como se cargan arreglos en memoria,si se utiliza macros,si se permite procedimientos recursivos,etc.

La estructura de datos,se refiere,a la forma como se almacenan los datos en una situación dada,por ejemplo, el formato que se usa,la información que se almacena (base de datos o banco de datos),etc.

Se necesita hacer estructura de datos,por que si tienen estos una estructura intrínseca y si se utiliza ésta para el procesamiento,hay mayor eficiencia para la ejecución; también si los datos tienen un tamaño variable y si se los estructura,hay un ahorro en la memoria utilizada,además de un acceso relativamente fácil.

Los macros o macroinstrucciones son abreviaciones de una cadena de procesos en un solo nombre previamente de-

finido,de acuerdo a una estructura convenida;es decir,se asignan nombres por ejemplo de cuatro caracteres para realizar algún conjunto de operaciones;por ejemplo,en el caso del Simulador,estas operaciones pueden ser:escribir en memoria y leer de memoria,etc.

También en la implementación en tiempo real,se incluye la administración de memoria;esta se organiza de acuerdo al lenguaje a utilizarse;habiendo una asignación en memoria de manera dinámica o estática.Por ejemplo en Fortran se tiene una asignación estática por medio de la sentencia DIMENSION.

En lo que tiene que ver con la optimización de los programas que constituirán el Simulador a realizarse,se cita aquí dos consideraciones principales a tomarse en cuenta:

-Eliminar operaciones redundantes sacando factor común de expresiones o parte de ellas repetidas;para ilustrar este caso se presenta aquí el siguiente ejemplo.

Ejemplo 2-5

Si se tiene la siguiente secuencia de operaciones en un programa :

1- D=D+C.B

2- A=D+C.B

3- C=D+C.B

En este ejemplo ,en la 1ra. y 2da. sentencia hay redundancia de la operación C.B,mientras en la 2da. y 3ra. hay repetición de la sentencia D+C.B

-Extracción de operaciones o expresiones constantes en lazos iterativos.Para visualizar esto,sea el siguiente ejemplo.

Ejemplo 2-6

Supon_gamos las siguientes líneas en un programa:

```
      DO 5  I=1,50
      A=3
5     X(I)=I+A
```

Aquí vemos que la sentencia A=3 puede ser realizada antes del lazo.

C A P I T U L O 3

PLANTEAMIENTO DEL SIMULADOR DEL MICROPROCESADOR M6800

En este capítulo se centraliza los criterios de Simulación descritos en los capítulos 1 y 2,en el objetivo de este estudio,que es,simular el microprocesador M6800; es así que luego de considerar ciertos detalles referentes al sistema donde vamos a simular y al código objeto proveniente del Cross Assembler,se hará un planteamiento del Simulador explicando este en base a submodelos,para cuya realización se tomarán en cuenta criterios del punto 1-2-2

Posteriormente,como mecanismos de control del Simulador,se plantearán comandos,o lo que es lo mismo,instrucciones dadas al Simulador para que este efectúe algunas operaciones.

Se cierra este capítulo con una enumeración de los posibles errores que podrán ser detectados por el Simulador.

3-1 CONSIDERACIONES DE LA CONFIGURACION GA-DM250:MEMORIA, ORGANIZACION EN DISCO,PERIFERICOS,SISTEMA CONVERSACIONAL, TRANSFERENCIA DE CODIGO OBJETO PROVENIENTE DEL CROSS-ASSEMBLER DEL M6800

3-1-1 SISTEMA GA-DM250

Es el sistema que se utiliza para la Simulación y es en donde tambíen están residentes los programas del Cross-Assembler del M6800.

Este sistema tiene 64K palabras de memoria real,de las cuales 32K palabras están disponibles para almacenamiento interno de acceso inmediato,esto se refiere a las posiciones de almacenamiento que el procesador central del DM-250 puede accesar en un tiempo mínimo.Además se dispone de otros medios de almacenamiento auxiliar tales como:Unidades de cinta magnética y Unidades de disco magnético,cuyo acceso involucra una demora y por lo tanto un tiempo mayor que para el caso de memoria real.

En un disco se dispone de 65 volúmenes cuya organización es por sectores,cada volúmen contiene 1576 sectores y cada sector contiene 320 palabras,siendo cada palabra de dos bytes;en el disco se leen o se escriben los datos por medio de las cabezas de lectura/gravación respectivos.

El minicomputador GA-DM250 tiene un sistema conversacional,caracterizado por la interacción entre el computador

y el programador,durante las diversas fases del desarrollo de sistemas,tales como:edición,compilación,depuración,ejecución,ingreso de datos y obtención de resultados.

El usuario en el terminal puede meter datos cuando el programa los requiera y ver la salida como se está produciendo.También el ususario puede parar la ejecución de un programa que no este dando los resultados deseados,puede dar diferentes parámetros de entrada,corregir errores de un programa y recompilarlo.

La información almacenada,representando datos o programas,residen en un aparato de almacenamiento directo.

3-1-2 CODIGO OBJETO DEL CROSS-ASSEMBLER DEL M6800

Descripción del funcionamiento.-

Al referirse a la fig. 3-1-2-1,se puede observar que cuando un usuario del Cross-Assembler[1],desea obtener a partir de un programa fuente en lenguaje Nmónico,un programa objeto en lenguaje de máquina,edita su programa y ensambla usando el Cross-Assembler(bloque C),obteniendo un programa objeto.

---

[1]El manejo del Cross-Assembler y su localización en el computador GA-DM250 se encuentran explicados en la Tesis del Cross-Assembler,indicada en la Bibliografía.

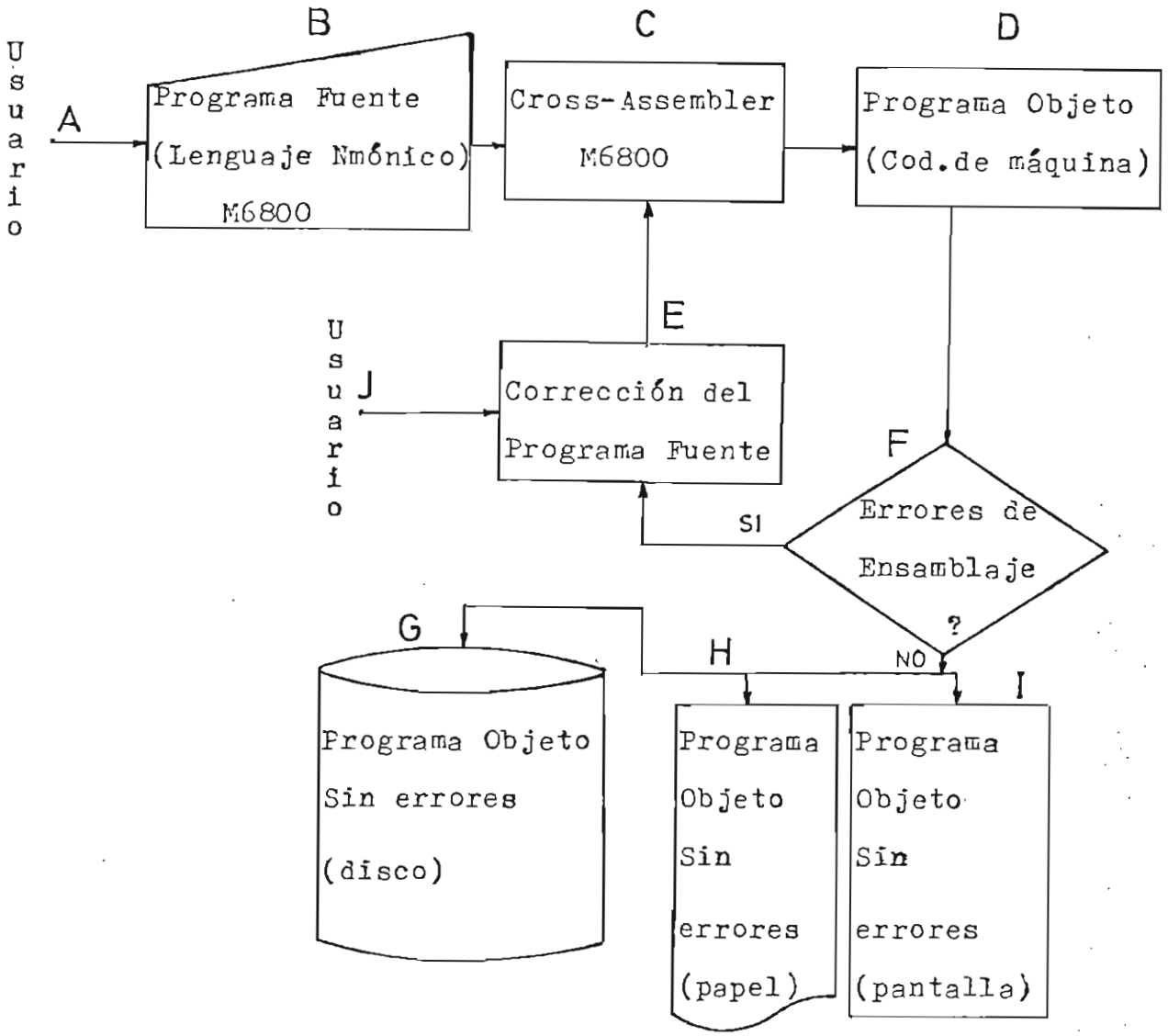


Fig.3-1-2-1 Código Objeto del Cross-Assembler del M6800

En esta figura,se observa también,que cuando un programa objeto(bloque D) contiene errores de ensamblaje,estos serían corregidos con la intervención del usuario que ensamblaría nuevamente,hasta cuando no se presenten errores;con lo que se obtendría un programa objeto sin errores de ensamblaje.El programa objeto en esta condición puede almacenarse en las siguientes formas:disco,papel o pantalla(bloques G,H,I respectivamente).

3-2 PLANTEAMIENTO ESQUEMATICO DEL SIMULADOR

3-2-1 Esquema Resumido del Simulador

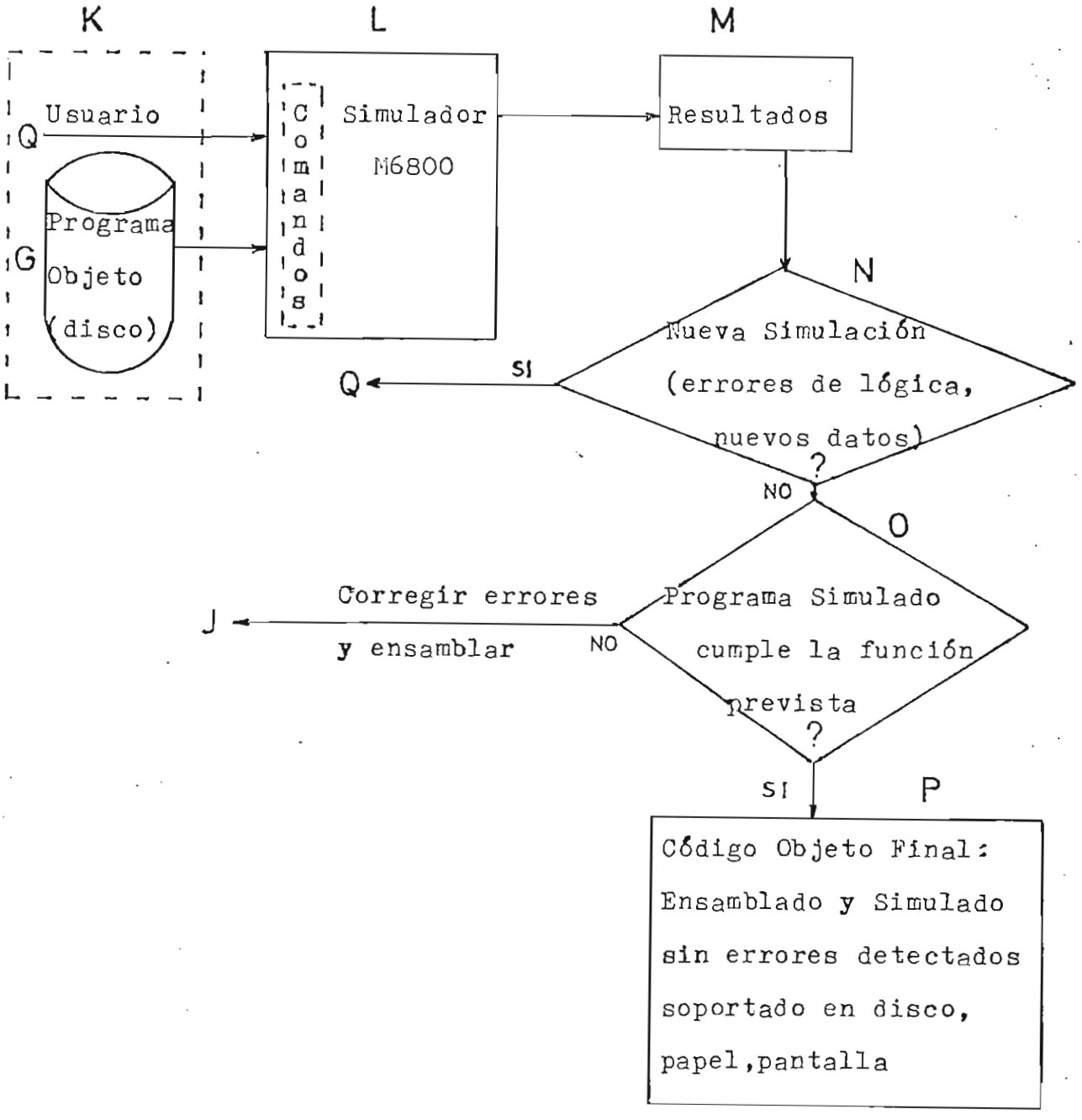


Fig.3-2-1-1 Simulador del M6800(esquema resumido)

En base a la figura 3-2-1-1 se analiza que:el programa objeto,sin errores,puede ser simulado,con la intervención del usuario por medio de comandos(bloque L).

Dependiendo del programa simulado y de los datos ingresados,se obtienen resultados,que pueden o no contener errores y de esta manera se requiere o no de una nueva simulación.

Además,se revisa, sí el programa cumple o no con la función prevista(bloque O).En el caso que no cumpla se corregirá errores y se ensamblará nuevamente.En caso contrario,cuando el objetivo del programa simulado es cumplido;se dispone ya de un código objeto final:ensamblado y simulado, sin errores detectados y que puede estar soportado en disco papel o pantalla(bloque P).

Para ilustrar la secuencia anterior,se incluye aquí el siguiente ejemplo:Sí se desea simular una suma del registro acumulador con una localidad de memoria;luego de realizarlo con cierto dato en la memoria,puede realizarse con otros en nuevas simulaciones.

Y,si analizando los resultados,se encuentra que el propósito del programa no era sumar,sino realizar por ejemplo,la función lógica OR entre un acumulador y una localidad de memoria,se deberá ensamblar y simular nuevamente.

3-2-2 CONSIDERACIONES DE LA APLICACION DEL CODIGO OBJETO ENSAMBLADO Y SIMULADO A SISTEMAS FISICOS

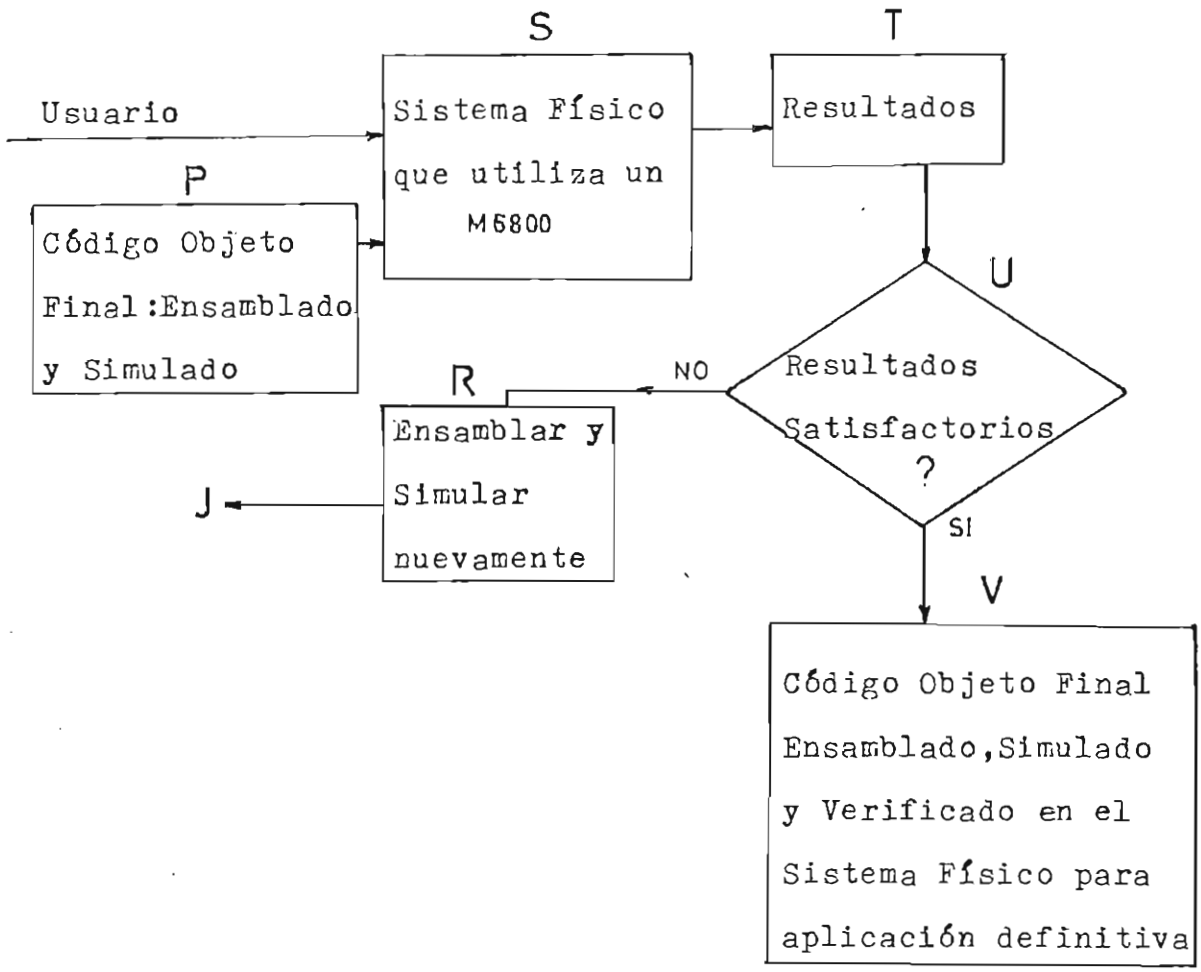


Fig. 3-2-2-1 Aplicación del Código objeto a Sistemas Físicos

Considerando la Fig.3-2-2-1,se observa que,para verificar el programa objeto ensamblado y simulado;se aplica el código objeto a un sistema físico que utilize el microprocesador M6800(este puede ser un Kit de desarrollo de sistemas basados en microprocesadores).Si los resultados obte-

nidos,no son satisfactorios,se pasará a corregir el programa fuente y a ensamblarlo nuevamente.Este procedimiento se repetirá hasta cuando se obtengan resultados esperados en el sistema físico;habiendose creado de esta forma,un código objeto final ensamblado,simulado y verificado.

3-2-3 DESARROLLO DEL ESQUEMA DEL SIMULADOR

Ampliando el diagrama de la fig.3-2-1-1 ,mediante un detalle del bloque L del mismo,se tendría;

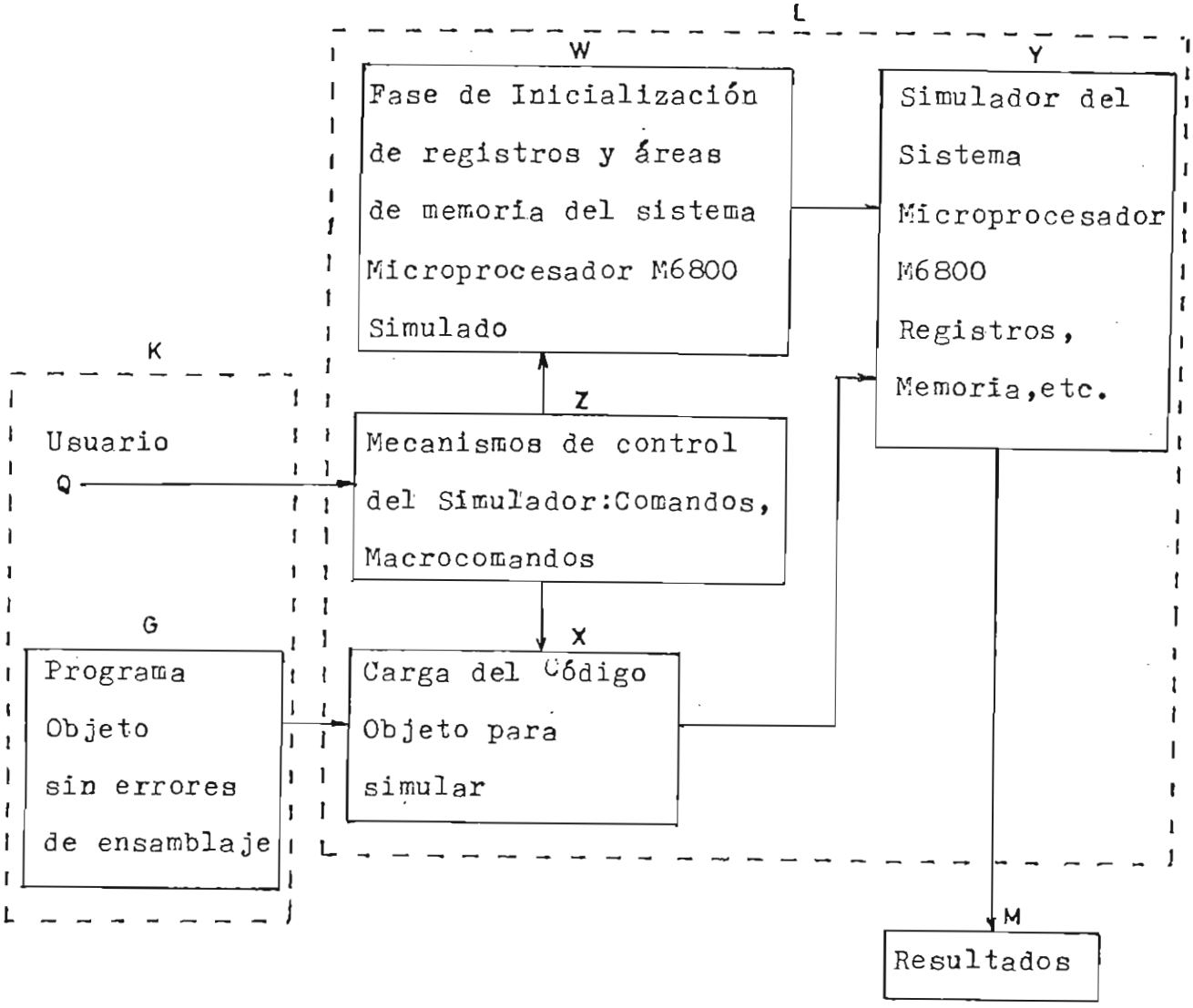


Fig.3-2-3-1 Acceso al Simulador del M6800

En la fig.3-2-3-1 se desarrolla la parte de acceso al Simulador,pudiendose ver que,antes de realizar la Simulación o Interpretación de un programa objeto,se hace una inicialización de registros,generalmente el encerado de estos(bloque W),las localidades de memoria utilizadas por el programa se inicializa con datos,etc.Por otro lado,el programa objeto es introducido en la memoria simulada(bloque X).

Siguiendo el desarrollo del Simulador,mediante un detalle del bloque Y de la fig.3-2-3-1,se obtiene el siguiente diagrama:

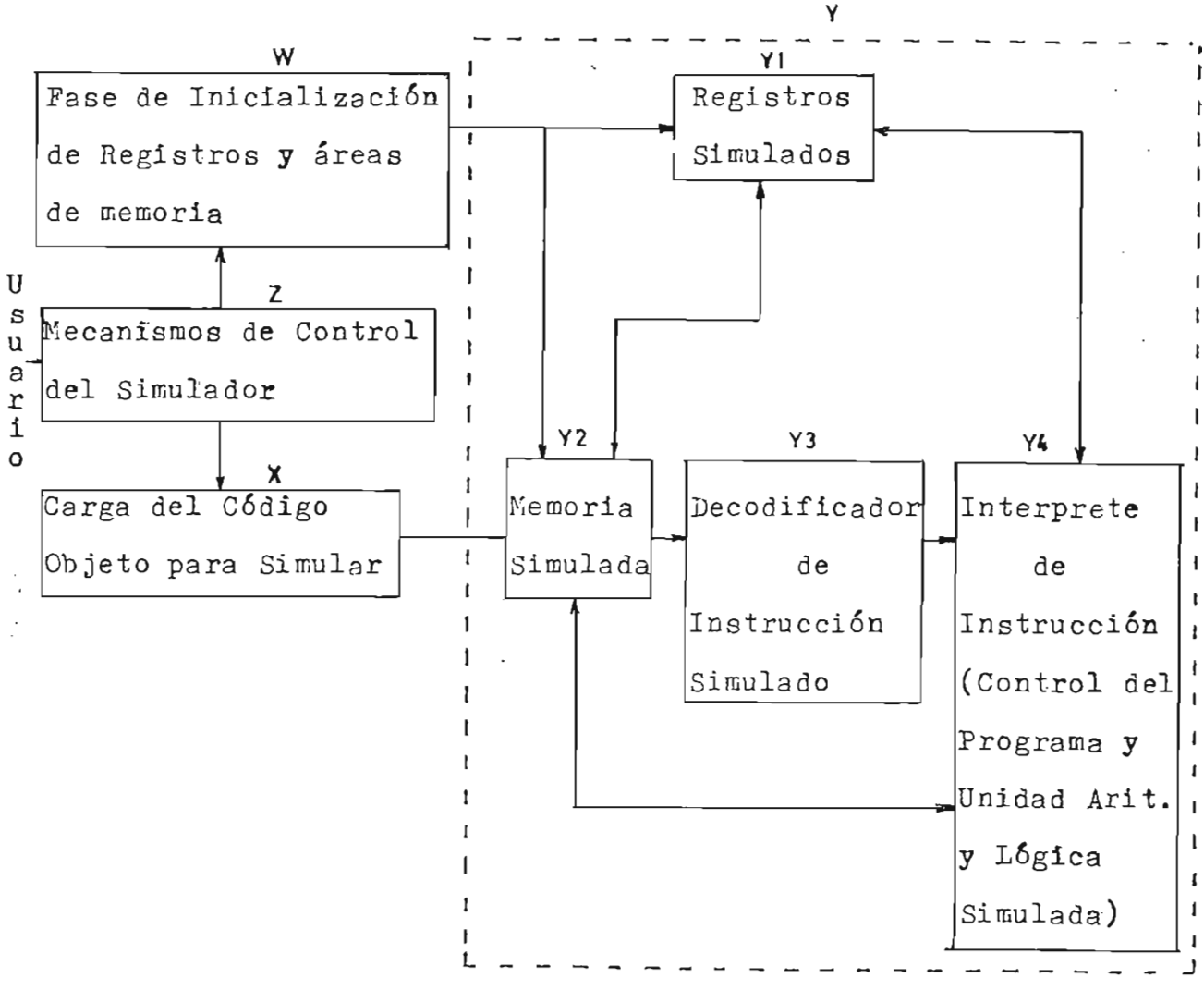


Fig.3-2-3-2 Simulador del M6800(bloques que lo constituyen)

3-3 DESCRIPCION DEL FUNCIONAMIENTO DEL SIMULADOR

Para describir el Simulador,se realizará la modelación de sus partes,es decir,se hará un submodelo de la memoria,un submodelo del decodificador de instrucciones,un submodelo de los registros y se describirá el intérprete de instrucciones que tendrá el control del programa simulado y simulará la función de la Unidad Aritmética y Lógica,ejecutando las instrucciones decodificadas.

Para realizar un modelo se tomará como base los criterios indicados en el punto 1-2-2.

3-3-1 SUBMODELO DE LA MEMORIA DEL M6800

Entidades u objetos de interés:el Microprocesador M6800 puede direccionar 65536 palabras de memoria,que en hexadecimal estarían entre las localidades 0000 hasta FFFF.

Atributos o propiedades de las entidades:cada palabra de memoria es de un byte,pudiendo almacenarse en esta,valores entre 00 a FF (en hexadecimal).

Entradas principales a memoria:Se tiene:direcciones de memoria,valor del registro contador del programa almacenado,valor del registro índice más offset en direccionamiento indexado; desde el intérprete se almacena en memoria resultados de operaciones con registros o con valores de memoria;otra entrada es la del código objeto del programa y también datos ini-

ciales para el programa.

Salidas de Memoria:instrucción direccionada o código del operador,tambíen el código del operando;áreas de datos.

Actividades o procesos que causan cambios en memoria:llegada de direcciones de memoria y datos a ser almacenados.

3-3-2 SUBMODELO DEL DECODIFICADOR DE INSTRUCCIONES

Entidades:Hay 197 códigos diferentes de instrucciones que utiliza el microprocesador M6800

Atributos:Existen 6 tipos de direccionamientodependiendo de la instrucción,el tamaño de cada código de instrucción es de 1 byte;se presenta diferente número de bytes para los diferentes tipos de instrucciones y sus operandos.Hay distintos tiempos en ciclos de máquina empleados en la ejecución de las diferentes instrucciones.

Entradas al Decodificador de Instrucciones:Instrucción direccionada en memoria.

Salidas del Decodificador:La decodificación de la instrucción es dada por un número de una subsentencia en el programa intérprete;este número es la salida del decodificador.

Actividades:LLegada de instrucciones al decodificador y busqueda del número de subsentencia,comparando secuencialmente

las instrucciones y obteniendo información del tipo de direccionamiento,tipo de registro con que trabaja la instrucción y el tiempo de ejecución para la instrucción.

3-3-3 SUBMODELO DE LOS REGISTROS DEL M6800

Entidades:Registros acumuladores A,B;registro contador del programa P,registro índice X,registro puntero del stack de memoria S,registro de condiciones C.

Atributos:Los registros A y B son de un byte,el registro C tiene 6 bits;los registros P,X,S tienen una longitud de 2 bytes.

Entradas:Los registros tienen como entradas,los valores dados en la inicialización del microprocesador simulado; pueden ingresar también valores desde el intérprete de instrucciones,donde algunos valores de registros cambian por efecto de las instrucciones y además pueden entrar valores de memoria a registros en instrucciones de carga de registros.

Salidas:Desde registros salen valores a memoria,por ejemplo el contenido del contador del programa para direccionar la siguiente instrucción en memoria,o valores de los registros a ser almacenados en memoria,o dirección del área de stack de memoria que se encuentra disponible y que se guarda en el registro S(stack pointer) y también salidas al intérprete de instrucciones,pues desde este se requieren valores de registros paralas operaciones que realiza el microprocesador simulado.

Actividades:El registro contador de programa varía con la secuencia que sigue el programa simulado,los registros acumuladores varían con las operaciones realizadas en el intérprete, al igual que los demás registros.

3-3-4 INTERPRETE DE INSTRUCCION DIRECCIONADA.-CONTROL DE PROGRAMA Y UNIDAD ARITMETICA Y LOGICA SIMULADOS

El control del programa se encuentra en el intérprete y utiliza las operaciones que le permite el computador simulador.

El submodelo de la Unidad Aritmética y Lógica son todas las operaciones aritméticas y lógicas que se describen en el set de instrucciones,como son por ejemplo:A+M→A;B+M→B,etc.

Este submodelo tendría como entidades las operaciones aritméticas binarias y decimales,las instrucciones lógicas de desplazamiento,las funciones de rotación,saltos,manipulación de stack,operaciones de transferencia de memoria,operaciones de interrupción;en total 72 operaciones diferentes.

Como atributos de este submodelo tenemos que algunas operaciones se realizan con acumuladores,otras con registros, otras con valores de memorias;hay operaciones para bits,para palabras de 1 byte y de 2 bytes.

Entradas para este submodelo son los contenidos de los registros,valores de memorias,información acerca de la ins-

trucción de la que se trata,como es:el número de bytes de esta instrucción,el número de ciclos de máquina que se demora en ejecutar esta instrucción,etc.

Como resultado de la intervención de la Unidad Aritmética y Lógica Simulada,se tiene eventualmente modificaciones en valores de registros,áreas de memoria,etc.;constituyendo las salidas en el submodelo.

Las actividades del submodelo de la Unidad Aritmética y Lógica son:sumas,restas,almacenamientos,cargas,rotaciones, saltos,etc.

3-4 COMANDOS DISPONIBLES[1]

Dentro del esquema del Simulador(fig.3-2-3-1)el bloque Z corresponde a mecanismos de control que son:comandos, macrocomandos y comandos a tabla de macrocomandos.A continuación se detallará todos estos.

3-4-1 COMANDOS PARA CONTROL DE FORMATOS E IMPRESION DE SALIDAS

-BE :Base de Entrada

---

[1]La descripción de los formatos de los comandos,está en el Manual de Usuario (Apéndice C)

Función:Este comando se usa para poner el control de formato del Simulador en la base que el usuario desee,pudiendo escoger entre:binaria,octal,hexadecimal o decimal.

-BS :Base de Salida

Función:Este comando seleccionará la base que el Simulador usará cuando imprima información para el usuario,pudiendo ser:binaria,octal,hexadecimal,decimal.

-SR :Selección de Registros

Función:Seleccionará cualquiera(o todos)los registros del M6800 para ser visualizados con el comando VR.

-DE :Da Encabezamiento

Función:Imprime la línea de rotulado de registros

-LC :Lista Comandos luego de ser introducidos

-NL :No lista comandos

-CO :Código Objeto

Función:Escoge la salida entre código objeto o código nmónico de las instrucciones que se vayan ejecutando.

3-4-2 COMANDOS PARA PONER VALORES

-PM :Poner en Memoria

Función:Se especificará una localidad de memoria con este comando y se pondrán valores a partir de esta localidad,de acuerdo a la base de entrada.Se puede utilizar este comando para cargar datos en ciertas localidades de memoria,requeridos por algún programa para el M6800,y/o para cargar un programa en lenguaje de máquina.

-PR :Poner en Registros

Función :De acuerdo a la base de entrada y especificando los registros a ser cambiados se ponen valores que tomarán estos. Los valores podrían ponerse en cualquier órden y unicamente los registros a ser cambiados necesitan ser ingresados.

-CM :Continúe poniendo en memoria

Función:Se utiliza a continuación de PM y este comando sirve para ir poniendo valores en las siguientes localidades de memoria a partir de la última localidad ingresada con PM.

-CP :Cargar el Programa

Función:Se utilizará este comando para cargar en memoria el código objeto ensamblado de un programa para el M6800;de esta manera luego de haberse transformado de lenguaje fuente a lenguaje objeto es cargado el programa objeto en el Simulador.

3-4-3 Visualización de valores

-VM :Ver Memoria

Función :Visualiza memorias desde cierta localidad inicial especificada y el número de localidades que se pida ser presentadas.Los valores de memoria visualizados están en la base seleccionada por el comando BS.

-VR :Ver Registros

Función :Con este comando podemos ver los registros seleccionados con el comando SR.Los valores numéricos de los registros aparecen en la base seleccionada por el comando BS.

-VU :Ver Ultima Instrucción

Función :Al ingresar este comando aparece la última instrucción ejecutada,siendo la salida en base hexadecimal.

-UD :Ultima Dirección

Función :Imprime la dirección de la última instrucción ejecutada en el programa,esta aparece en la base especificada por BS.

-ER :Mensaje de Error

Función :ingresando en este comando el número de error,aparece el mensaje de este error.

3-4-4 Comandos de Ejecución de Programa

-EJ:Ejecuta el Programa

Función :Especificando la localidad donde empieza el programa y el número de pasos que se desea que se ejecuten (de acuerdo a la base de entrada), se ejecuta el programa; para luego imprimir todos los registros en base hexadecimal.

-RN :Rastree N pasos del Programa

Función :Indicando en este comando la localidad donde se inicia el restreo y el número de pasos que se requiere rastrear, este comando luego de ejecutar cada instrucción imprime todos los registros en base hexadecimal.

-RS :Rastree N saltos de programa

Función :En un programa que contenga saltos, luego de cada salto se imprime todos los registros en base hexadecimal.

-EI :Ejecuta Instrucciones

Función :Cumple igual función que EJ, pero aquí se ejecuta un número de instrucciones especificado.

3-4-5 Comandos de Interrupción Simulada

-IR :Interrupción de hardware simulada

Función :Utilizado este comando luego del ingreso de una

instrucción WAI (Espera por interrupción), se pasa al programa de interrupción que se inicia la dirección de memoria almacenada en las localidades FFF8,FFF9; revisando previamente que la bandera de interrupción no este seteada; mediante este comando se simula la señal IRQ. ("Interrupt request")

-IS :Interrupción sin máscara

Función :También se utiliza luego de una instrucción WAI; sin tomar en cuenta la bandera de interrupción, pasa al programa de interrupción que se inicia en la dirección de memoria almacenada en las localidades FFFC y FFFD. Mediante este comando se simula la señal NMI("Non-Maskable Interrupt")

3-4-6 Macrocomandos

Un macrocomando es una abreviación de una serie de comandos del Simulador, los cuales son realizados en el orden definido, cada vez que el nombre del macro es llamado.

Un macro incluye una tira de comandos que en un determinado orden se desean repetir, ahorrando el tiempo y esfuerzo de reentrar estos comandos cada vez que ellos sean requeridos.

Para ejecutar un grupo de comandos solamente se ingresa el nombre del macro asociado.

El macro al definirse se guarda en una tabla de macros para posteriormente ser llamado durante similares y posteriores simulaciones.

La idea de un macro es la inserción o sustitución de texto, un identificador o nombre es sustituído por una tira de caracteres en este caso comandos.

Ejemplo:

Definición de Macro		Simulación	Resultado
MAC	BSO2.VMm,n	MAC	BSO2 VMm,n

En este ejemplo se define con el nombre MAC la tira de comandos :base de salida binaria, ver n localidades de memoria desde la localidad m; el resultado es la visualización en binario de las memorias.

El nombre MAC cuando aparece es una llamada a macro; el proceso de sustituir la llamada de macro por su cuerpo se conoce como expansión de macro.

Una llamada a macro[1] se expansiona así :la tabla de macros (conteniendo nombres de macros y sus tiras asociadas) se explora para tomar uno por uno los integrantes de la definición del Macro.

---

[1]La forma de generar macro comandos,asi como, otras características de estos; está explicado en el Manual de Usuario (Apéndice C).

La forma de definición de macros es así:

Nombre (tira)

La tira es una secuencia de comandos que pueden ir separados por un punto por ejemplo, además el nombre se limita a una cierta extensión de caracteres (4 por ejemplo) y que se empieze por un caracter alfabético.

La definición del macrocomando (tira) se encierra entre paréntesis.

Ejemplo :

Si se quiere realizar la siguiente secuencia de comandos:

-Poner Base de salida en 16

-Seleccionar los registros : A,B,X

-Visualizar los registros

Utilizando algún nombre de macrocomando,por ejemplo ALFA,la definición se hará así:

ALFA (BS16.SR ABX.VR)

La tira de comandos se pone en efecto poniendo únicamente el nombre del macrocomando:

ALFA

3-4-7 COMANDOS A TABLA DE MACROCOMANDOS

-LM:Liste Macros

Función:Presenta una lista de los macrocomandos almacenados, pudiendose ver qué capacidad remanente hay para guardar más macros,considerando que se dá un límite de macros a almacenarse;este límite,de acuerdo a la memoria se escogerá posteriormente.

-BM:Borre Macrocomandos

Función:Se tiene la posibilidad con este comando de ir dando nombres de macros que se desean borrar,ya sea por que no se van a utilizar o se quiere dar espacio para otros macros.Se borra macros en caso de tener almacenado un número de estos cercano al límite a guardarse.

3-5 ERRORES QUE DETECTARA EL SIMULADOR[1]

El Simulador detectará errores en comandos no ingresados correctamente,en macrocomandos mal definidos,en comandos inexistentes,en instrucciones del microprocesador inexistentes y en general cualquier condición de mal uso o uso no

---

[1]Ver punto 3-4-3 comando ER,sobre la visualización de mensajes de error.

previsto del Simulador.

ERROR 1 :De Sintaxis en Macrocomando

Significado :Error de sintaxis en nombre de macrocomando; se acepta letras como primer caracter del nombre y letras o números en los siguientes;debiendo ser el nombre de cuatro caracteres.

ERROR 2 :En definición de Macrocomando

Significado :No está balanceado los paréntesis o hay espacios en blanco luego de puntos o longitud de caracteres es mayor de 60.

ERROR 3 :Librería de Macros llena

Significado :Se llegó al límite de almacenamiento de macrocomandos;por lo tanto el macro no fue almacenado.Se da un límite de 29 macrocomandos a ser almacenados en la tabla.

ERROR 4 :Se quiere crear un macrocomando existente

Significado :Se quiere dar un nombre de macrocomando que ya existe.En caso de que se quiera ejecutar el macrocomando debe ponerse unicamente el nombre del mismo.

ERROR 5 :No existe el macrocomando que se pide borrar

Significado :Se ha dado instrucción de borrar un macrocoman-

do que no existe en librería de macros.Sí se desea revisar los macrocomandos almacenados utilizar el comando LM.

ERROR 6 :Error en nombre de macrocomando a borrar

Significado :El nombre de un macrocomando que se pide borrar es mayor que 4 caracteres,o empieza con un caracter distinto a una letra.

ERROR 7 :Un comando no definido en el Simulador fue encontrado

Significado :Solo se pueden utilizar los comandos descritos en el punto 3-4 y con el formato indicado en el Manual de Usuario (Apéndice C).También el error puede ser por haber utilizado un comando CM sin antes haberse ejecutado un comando PM.

ERROR 8 :Error en comando del Macro

Significado :El comando que está dentro del macrocomando no ha sido definido;esto es detectado cuando se va a ejecutar un macrocomando y al ir tomando consecutivamente cada comando se encuentra un comando que no corresponde a los realizados en este Simulador.

ERROR 9 :Direccionamiento Ilegal

Significado :Una dirección ilegal de memoria fue encontrada, la dirección no está dentro del rango de memoria simulada (en hexadecimal de 0000 a FFFF)

ERROR 10 :Valor de entrada sobrepasa la capacidad

Significado :El valor de entrada,si es a memoria ha sobrepasado un byte (FF en hexadecimal) y si es un valor a un registro depende de este,si es acumulador A o B no debe sobrepasar FF en hexadecimal y si es X por ejemplo no debe sobrepasar (FFFF en hexadecimal).

ERROR 11 :No se Simula

Significado :No se puede simular un programa que luego de ser ensamblado,tiene algún error detectado en el ensamblaje.

ERROR 12 :Error en operando

Significado :Error en instrucción.El Simulador encontró uno de los 59 códigos no definidos , es decir,que no constituye una instrucción válida del microprocesador.

ERROR 13 :Error en comando de interrupción

Significado :En una interrupción simulada solo se pueden ingresar los comandos que representan interrupción que son:IR o IS.

Entre los resultados de un rastreo de un programa por medio del comando RN se observará como el contenido de los acumuladores, registro índice, stack pinter y registro de tiempo cambian; pudiendo ser chequeados antes y después de cada instrucción ejecutada.

La dirección de la instrucción donde empieza la ejecución de un programa, es ingresada en los comandos de ejecución: EJ, RN, RS, o EI ; y en los resultados podemos ver como avanza el programa en el número de bytes requeridos para cada instrucción.

El contador del programa irá indicando donde está localizada la próxima instrucción. También será interesante visualizar como el registro de condiciones muestra el efecto de las operaciones del acumulador, trnsferencia de datos de registros, rotaciones y el estado que presenta para saltos condicionales.

Se dispondrá de un registro T (Tiempo) que nos indique los ciclos de microprocesador que se van aumentando como efecto o resultado de la ejecución de cada instrucción.

De acuerdo a como se van dando los resultados, el stack pointer y el registro índice deberían ser revisados para estar seguros que no hay superposición con memorias reservadas para otras instrucciones en el programa o parámetros.

Las memorias para almacenamiento temporal de datos pueden ser examinadas en diversos instantes de la ejecución

de un programa,mediante corridas parciales;para estar seguros que los valores esperados están siendo obtenidos.

C A P I T U L O 4

IMPLEMENTACION DEL SIMULADOR DEL MICROPROCESADOR M6800

Luego de haberse planteado al Simulador en el Capítulo anterior;se pretende en este Capítulo realizar su implementación ,para lo cual se usará el lenguaje Fortran,cuya utilización fue justificada en el punto 1-5.

La implementación del Simulador tomará como base la descripción del Sistema realizada en 3-2;donde fue presentado el Sistema mediante bloques y luego analizado el mismo por medio de submodelos en 3-3.

Para la programación del Sistema Simulador se ha utilizado archivos,subrutinas,programas principales.En cierto caso fue necesario utilizar los criterios de máquinas de estado finito (ref. 2-2)

Para visualizar la secuencia en los programas y subrutinas,se dará a conocer los respectivos diagramas de flujo, siendo estos además de gran utilidad para proporcionar información del funcionamiento,conformación,análisis,futuros cambios,etc;de los programas implementados.

4-1 IMPLEMENTACION DEL SIMULADOR EN FORTRAN . ARCHIVOS REALIZADOS

En esta sección se toman en cuenta algunas consideraciones de la programación en Fortran, específicamente las que se necesitarán en la Simulación del microprocesador M6800 ; además por requerirse el gravar cierta información en los denominados archivos; se detallará la forma de realizarlos y se describirá los implementados.

Previamente a la implementación del Simulador, se incluirán ciertos criterios definiendo el sistema microprocesador que se simula, en una forma global.

4-1-1 DEFINICION DEL SISTEMA

Es necesario especificar en esta parte lo que abarcará la simulación del M6800, pues cuando se realizó la descripción del Simulador por medio de submodelos se había analizado de una manera parcial.

Para el propósito del estudio que se están realizando, la simulación comprende : los registros del microprocesador M6800, la memoria que puede direccionar, el decodificador de instrucciones y la Unidad Aritmética y Lógica ; excluyendose las conexiones exteriores, la alimentación del microprocesador, las características físicas, etc.

Basándose en los conceptos de Simulación expuestos

en 1-2 ,se puede afirmar que el Sistema microprocesador a simularse es un Sistema Discreto por que tiene cambios discontinuos;es un Sistema Abierto pues acepta información desde el exterior del Sistema ,en forma de datos o programas.

El ambiente del Sistema ,es decir, todo lo que está fuera del mismo,en el caso del microprocesador a simularse, correspondería a las conexiones exteriores como : bus de datos,bus de direcciones,señales de control de tiempo,señales de control de los buses,señales de control de la memoria y señales de interrupción ; todas estas no serán consideradas en la simulación ,y únicamente las señales de interrupción NMI e IRQ serán simuladas luego de una instrucción WAI.

4-1-2 CONSIDERACIONES DE LA IMPLEMENTACION EN FORTRAN.-

Se verá a continuación como trabajará el Simulador a implementarse y las especificaciones que en Fortran han sido necesarias utilizar.

El intérprete y el programa simulado estarán en la memoria del computador a utilizarse.Al comienzo de la Simulación se proporcionará al intérprete la posición de partida del programa ; esta posición se colocará en un registro contador de programa simulado ,se interpretará o ejecutará la instrucción de partida , mientras el contador de programa simulado indicará la siguiente instrucción a ejecutarse.Cuando se produzca una transferencia de control ,la dirección adecuada se colocará en el registro contador de pro-

grama simulado y cuando no haya transferencia el contenido de este registro se aumentará en:uno,dos o tres de acuerdo al número de bytes que tenga cada instrucción.

A lo largo de la Simulación ,el control permanecerá con el intérprete y el registro contador del programa simulado guiará la ejecución de este programa; cada operación se ejecutará mandando el control a una sentencia adecuada dentro del intérprete.

En los programas en Fortran se utilizará las especificaciones : TWO WORD INTEGERS y EXTENDED PRECISION ,es decir,enteros de dos palabras (4 bytes) y reales de 6 palabras (12 bytes) .Se debe limitar para ciertos registros simulados a su capacidad respectiva,por ejemplo enteros de un byte,lo mismo que para valores a memoria y enteros de dos bytes para otros registros;por lo tanto se necesitará escribir subrutinas que luego de operaciones con los registros sean llamadas para limitarlos a los valores que en el microprocesador no puedan sobrepasar.

De acuerdo a los comandos especificados en el punto 3-4 ,se deben introducir datos al Simulador en las distintas bases : binaria,octal,hexadecimal,decimal ; lo mismo que se va a necesitar sacar valores del Simulador y darles una presentación en cualquiera de las bases indicadas ; por lo tanto se hará necesario escribir subrutinas que transformen de base binaria a decimal,de octal a decimal,de hexadecimal a decimal y subrutinas para imprimir valores,es decir,de de-

cimal a binaria,de decimal a hexadecimal,de decimal a octal; y esto tomando en cuenta que todas las operaciones internas del Simulador se harán en decimal.

Otra consideración que hay que hacer es que en Fortran no existen instrucciones para trabajar con bits y por necesitarse en el Simulador direccionar bits en algunas operaciones,se requerirán subrutinas que a una palabra que representa un registro en decimal se pase a 8 o 16 bits ,con una palabra para cada bit.

4-1-3 ARCHIVOS IMPLEMENTADOS.-

En la Simulación se utilizarán archivos para guardar grupos de datos,teniendo estos las siguientes funciones: simular la memoria,guardar los códigos del microprocesador, gravar los comandos disponibles,asegurar la librería de macrocomandos,archivar los errores que se detectarán,mantener los códigos mnemónicos y de máquina correspondientes.

En Fortran los archivos se implementan así:

Definición : DEFINE FILE A(X,Y,U,J)

donde :

	A	Número del archivo
	X	Número de registros contenidos en el archivo.
	Y	Número de palabras por registro.

U Archivo sin formato

J Iuntero del archivo.

Localización :La localización de un archivo se hace utilizando un comando del Sistema,así:

ALLOCATE (NOMBRE DEL ARCHIVO)VOLUMEN TAMAÑO

Este comando reserva espacio en disco,en el volúmén indicado,siendo el tamaño en sectores y considerando que cada sector=320 palabras ,el número de sectores de un archivo se calcula así:

$$\#\ \text{sectores} = \frac{\#\ \text{Registros} \times \#\ \text{palabras/registro}}{320\ \text{palabras/sector}}$$

ARCHIVO 1 : MEMRO

Localización : ALLOCATE MEMRO.S(4EE1) SECTORS 415

Definición : DEFINE FILE 1 (66000,2,U,J1)

Objetivo : Este archivo simula la memoria que puede direccionar el microprocesador M6800.

La simulación de memoria se hace de acuerdo al submodelo de memoria explicado en 3-3-1.El número de registros son 65536,sin embargo en el archivo se sobredimensiona a 66000 registros.Cada registro tiene un byte ;en este archivo también se ha aumentado este valor a 4 bytes que hay en dos palabras, y esto es debido a que los programas y subrutinas se realizarán con la especificación de enteros de dos palabras.

Número de sectores :

$$\#\ \text{sectores} = \frac{66000 \times 2}{320} = 412 \rightarrow 415$$

ARCHIVO 2 : CO2P

Localización : ALLOCATE CO2P.S(4EE2) SECTORS 10

Definición : DEFINE FILE 2 (256,10,U,J2)

Objetivo y Diseño del archivo : Guardar las instrucciones del M6800

De acuerdo al esquema del Simulador,para el decodificador de instrucciones,se necesita tener almacenado en un archivo las instrucciones del M6800;además en base a lo descrito en el submodelo del Decodificador de instrucciones en 3-3-2,se debe guardar información de las instrucciones;lo que se ha hecho en este archivo de la siguiente manera:

-Código de instrucción : 11,12,....,64 ;se utilizarán estos códigos para direccionar las subsentencias en el programa intérprete,siendo el mismo código para instrucciones que realizan la misma operación.

-Código de direccionamiento :

1 Direccionamiento Inmediato

2 Direccionamiento Directo

3 Direccionamiento Indexado

4	Direccionamiento Extendido
5	Direccionamiento Inherente
6	Direccionamiento Relativo

-Tipo de Registro : Dentro de un mismo código de operación se asignan números que distinguen a los registros que realizan esa operación, así por ejemplo para la operación :

ADD	Se tiene :	1	ADDA
		2	ADDB
		3	ABA

-Tiempo :Número de ciclos de máquina para cada instrucción.

Por lo tanto, este archivo se conforma así:

OPERANDO	CODIGO	DIRECCIONAMIENTO	REGISTRO	TIEMPO

Número de sectores :

$$\#\ \text{Sectores} = \frac{256 \times 10}{320} = 8 \rightarrow 10$$

ARCHIVO 3 : REGIS

Localización : ALLOCATE REGIS.S(4EE1) SECTORS 1

Definición : DEFINE FILE 3 (1,20,U,J3)

Objetivo : Guardar los registros simulados del microprocesador.

El mantener los registros en un archivo,permite asegurar el valor de los registros luego de las operaciones,inicializar registros antes de la ejecución de un programa,revisar los registros en que valor han quedado por medio del comando VR (Ver registros),etc.

Amás de los registros del microprocesador M6800,se guarda también :la dirección de la instrucción que se ha procesado (I),dirección efectiva del operando (E) y el registro de tiempo (T) que contiene el número de ciclos de máquina.

El archivo está organizado así:

I	O	P	E	X	A	B	HINZVC	S	T

donde : I Dirección de la instrucción

O Código del operador

P Contador del programa

E	Dirección efectiva del operando
X	Registro índice simulado
A	Acumulador A simulado
B	Acumulador B simulado
HINZVC	Banderas de condición.
S	Stack Pointer
T	Tiempo

Número de sectores :

$$\#\ \text{Sectores} = \frac{1 \times 20}{320} \rightarrow 1$$

ARCHIVO 4 : (Nombre del programa ensamblado)

Localización : Este archivo se localiza al editar el programa fuente en el volúmen asignado para editar en el Cross-Assembler,siendo el 4005 o el 4006; si se le llama al programa a ensamblar PRGM,la localización se haría así:

ALLOCATE PRGM.S(4006) SECTORS ($\frac{\#\ \text{líneas de prog.}}{2}$ + 1)

Definición : DEFINE FILE 4 (1000,160,U,J4)

Objetivo : Guardar un programa fuente y su código objeto.

Este archivo contiene un programa en lenguaje fuente y luego de ensamblarlo contiene también el programa en lenguaje objeto.

De este archivo y mediante el comando del Simulador CP se toma el código objeto y se grava en la memoria simulada (se gravará en memoria siempre que sea un código objeto sin errores detectados en ensamblaje).

ARCHIVO 5 : COM

Localización : ALLOCATE COM.S(4EE2) SECTORS 1

Definición : DEFINE FILE 5 (22,12,U,J5)

Objetivo : Almacenar los comandos del Simulador.

Debido a que se utilizarán mecanismos de control del Simulador como son: comandos,macrocomandos; que se especifican en el esquema general del punto 3-3-4 y explicados en 3-4, se requiere tener guardados los comandos en un archivo.

Nomenclatura del Archivo :

-Comando : El archivo contendrá todos los comandos descritos en 3-4.

-Código : Para cada comando se da un código que servirá para direccionar una subsentencia ,en el programa que ejecuta cada comando.

-Especificación : En los comandos se especifican características que ayudan en la ejecución de comandos;por ejemplo para BE y BS ,este número representa el valor de la base.

-Propiedad 1 : Se utiliza para los comandos BE y BS ,y guarda el número de caracteres que hay en una palabra de dos bytes en la base especificada.

-Propiedad 2 : Se utiliza para los comandos BS y BE ; y es un código para distinguir las bases así:

Base : 2	Prop.2 : 1
8	2
16	3
10	4

Cada registro en este archivo se presenta así:

COMANDO	CODIGO	ESPECIFICACION	PROP.1	PROP.2

Número de sectores :

$$\#\ \text{Sectores} = \frac{22 \times 12}{320} = \rightarrow \quad 1$$

ARCHIVO 6 : MACRO

Localización : ALLOCATE MACRO.S(4EE1) SECTORS 16

Definición : DEFINE FILE 6 (30,150,U,J6)

Objetivo : Guardar los macrocomandos implementados.

Se gravarán los macrocomandos de acuerdo a la forma de definición de estos,descrita en 3-4; es decir,se asigna una extensión de 60 caracteres para el macrocomando ,pudiendo ser el nombre de 4 caracteres máximo y empezando con un caracter alfabético.Luego del nombre hay un espacio en blanco y a continuación está la definición o tira de comandos entre paréntesis.Los comandos están separados por puntos y no debe haber espacios en blanco.

Al guardarse el macrocomando en el archivo,el programa busca las posiciones de puntos y paréntesis y las pone a continuación del macrocomando;esto servirá para ir tomando comando por comando en la ejecución de un macro.

Se asigna un límite de 30 registros para este archivo,es decir,que habrá una capacidad para 29 macrocomandos ya que el 1er. registro tiene la cuenta de macros.

Este archivo puede ser visualizado por medio del comando LM (Liste Macros).

En los nombres de macrocomandos,las dos primeras letras no deben ser iguales a las de un comando;pues los comandos tienen prioridad sobre los macrocomandos.

Un ejemplo de macrocomando puede ser :

ABC (BE16.FR I0000,A00,B00)

El comando se guardará de la columna 1 a la 60.

1 . . 60	Puntos		Paréntesis	
ABC ( )	10		5	27

Número de sectores :

$$\#\ \text{Sectores} = \frac{30 \times 150}{320} = 14 \rightarrow 16$$

ARCHIVO 7 : ERR

Localización : ALLOCATE ERR.S(4EE1) SECTORS 25

Definición : DEFINE FILE 7 (50,160,U,J7)

Objetivo : Guardar los errores que puede guardar el Simulador.

Los errores de este archivo podrán ser leídos con el comando ER.

Este archivo de errores se organiza así : Empezando en el registro 2 se graba números de registros donde se inicia y término el 1er. error ; en el registro 3 los del 2do. error y así hasta el número de errores que se quiera grabar que es máximo de 14 y a partir del registro 15 se guarda los enunciados de los errores.

El archivo se mostraría así:

Registro	Inicio	Final
2	15	17
3	18	18
--	--	--
15	Enunciado del Error 1	
17		

Número de sectores :

$$\#\ \text{Sectores} = \frac{50 \times 160}{320} = 25$$

ARCHIVO 8 : NMONI

La localización y definición de este archivo se encuentran explicados en la descripción de archivos del Cross-Assembler (Referirse a la Tesis indicada en la Bibliografía);

Este archivo guarda los códigos nmónicos delas instrucciones del M6800 , con los correspondientes códigos de máquina.

4-2 DIAGRAMAS DE BLOQUES DEL SISTEMA

Considerando que los diagramas permiten representar de un modo gráfico y ordenado las operaciones a realizarse en los procesos y que por medio de los diagramas de bloques se puede identificar en forma general un determinado proceso, se los utilizará para describir la programación del Simulador.

Para los diagramas a realizarse se tomarán en cuenta los esquemas planteados en 3-2.

4-2-1 OBJETIVOS DE LOS DIAGRAMAS

-Establecer una visión mucho más amplia de un Sistema, así como también realizar una verificación de haberse tomado en cuenta todas las posibilidades en ese Sistema.

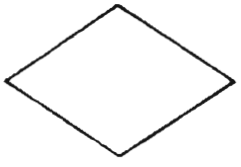
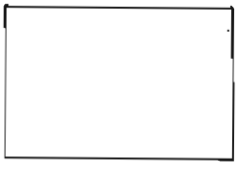
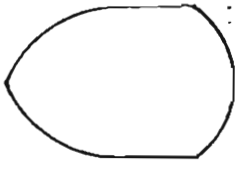
-La solución de algunos problemas por medio de diagramas es otro objetivo, ya que puede ser más fácil hacerla representando gráficamente.

-Facilitar las modificaciones en programas, enfocandose directamente ciertas partes que necesitan ser cambiadas.

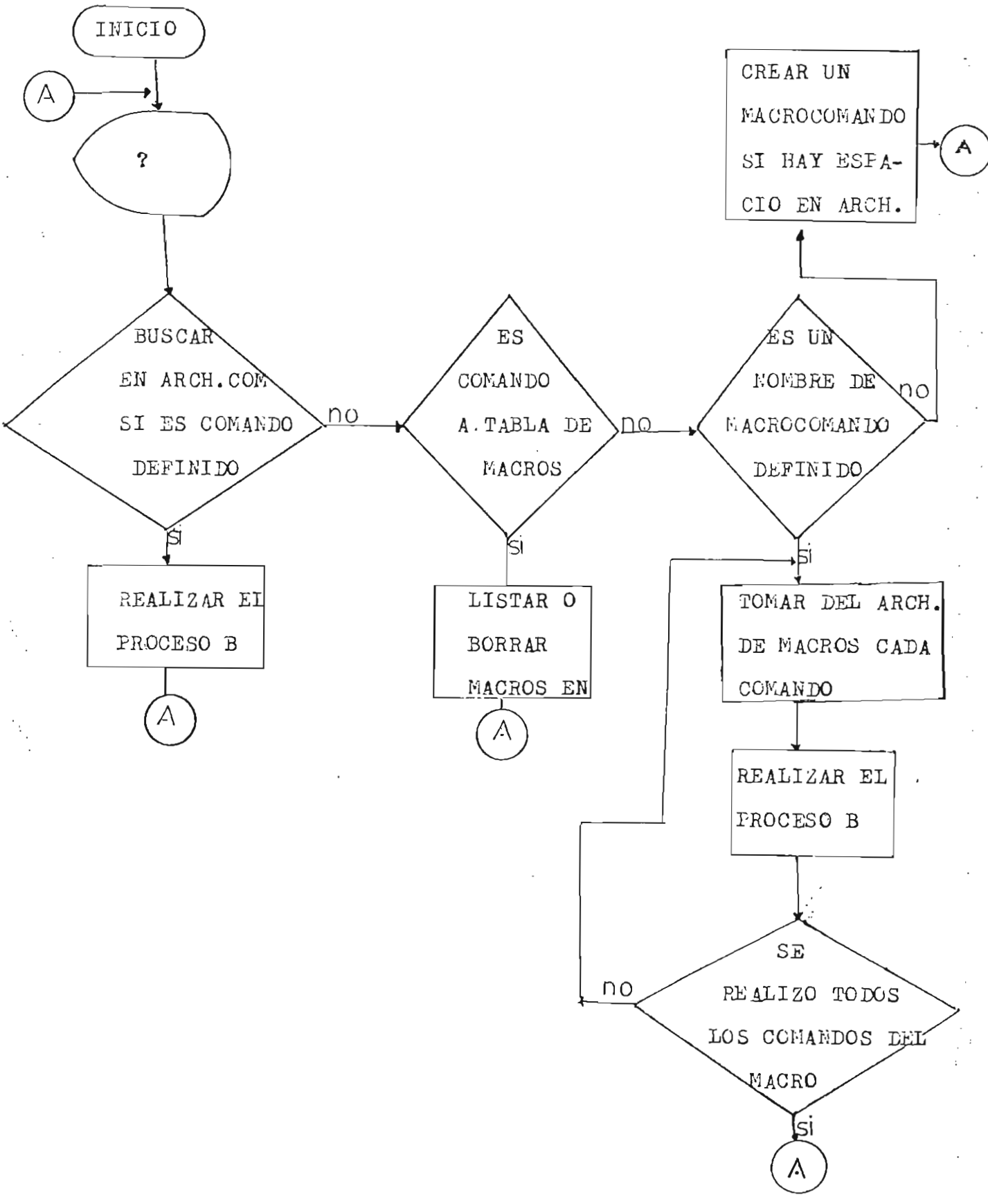
-Ayudar a una codificación efectiva y rápida.

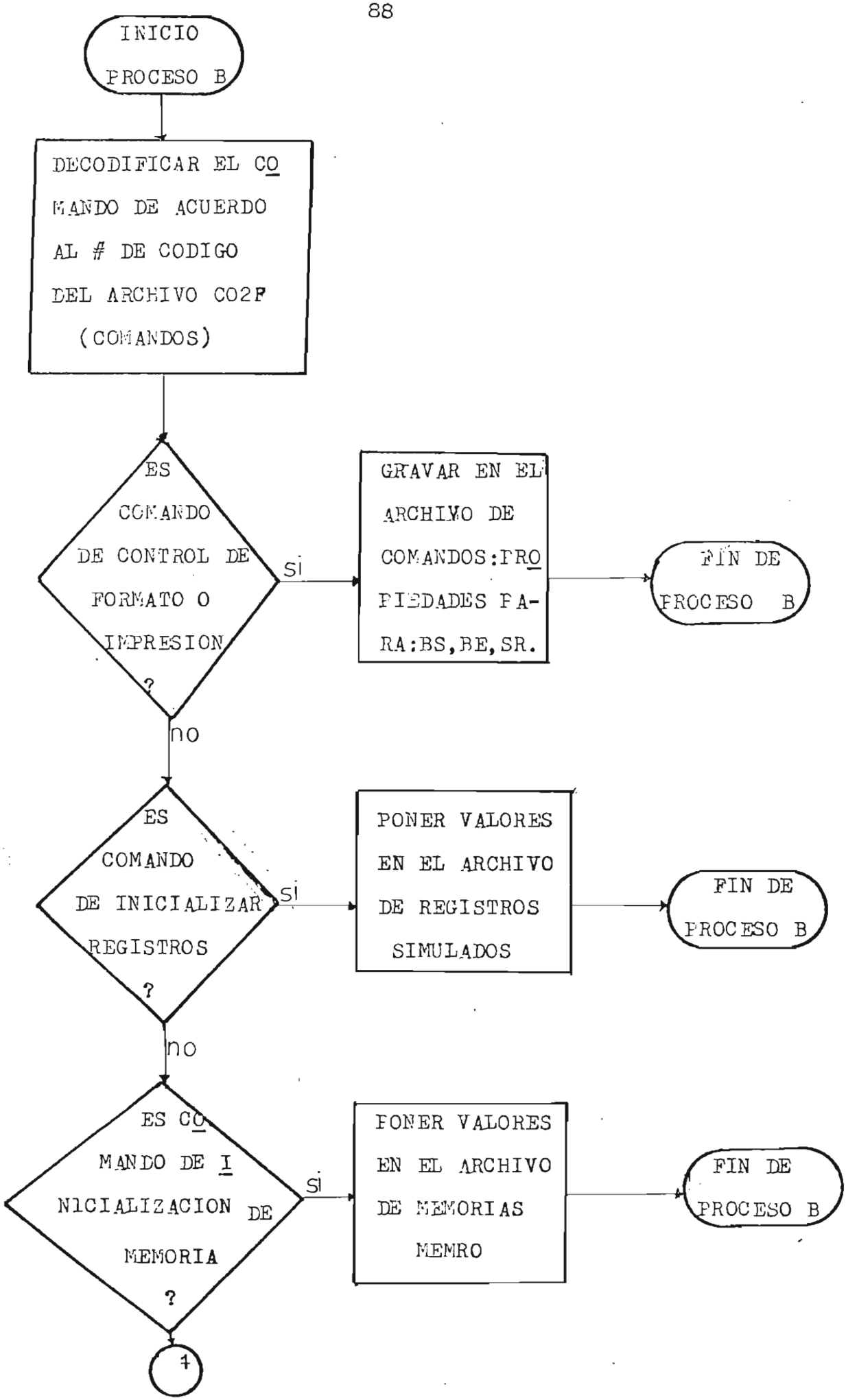
-Documentar gráficamente un programa, aumentándose la comprensión de los mismos.

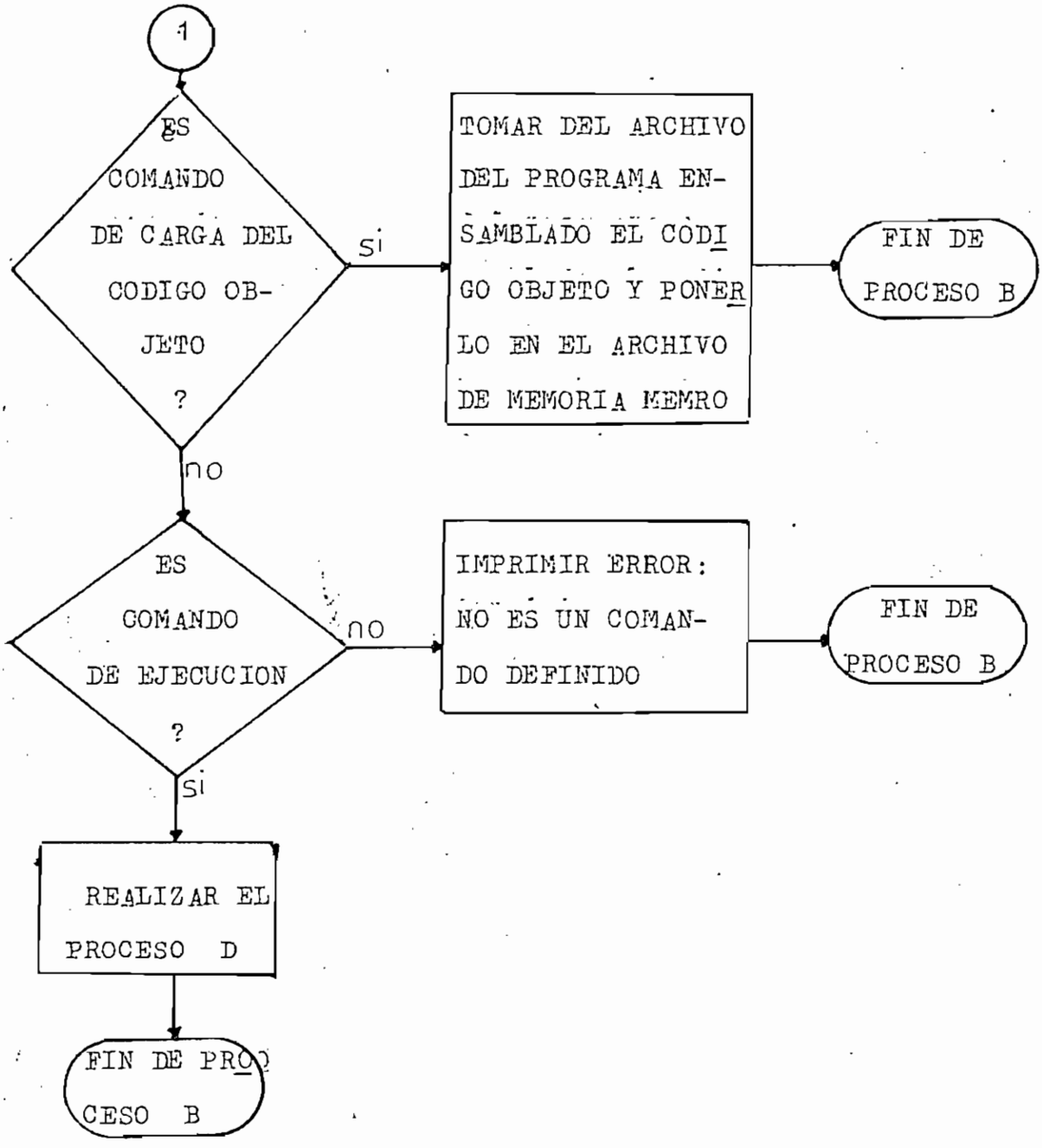
4-2-2 SIMBOLOGIA PARA LOS DIAGRAMAS DE BLOQUES

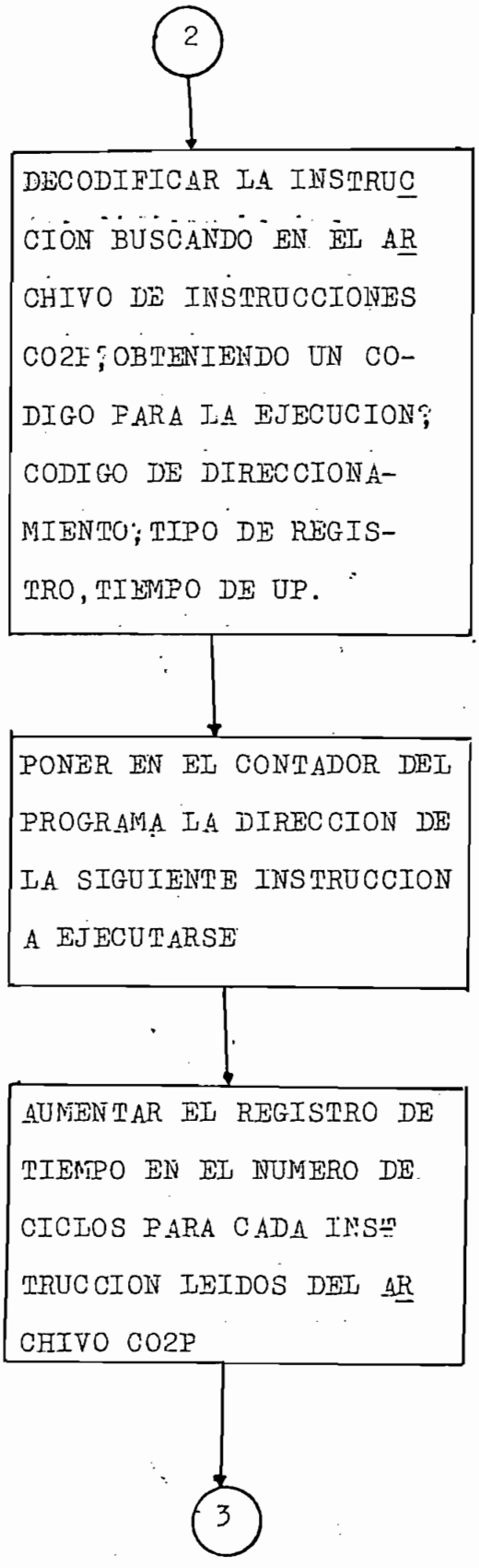
SIMBOLO	SIGNIFICADO
	Principio/Fin de un proceso
	Toma de decisión que determina caminos alternativos
	Proceso en general
	Comprobación visual (DISPLAY)
	Conectores entre Símbolos

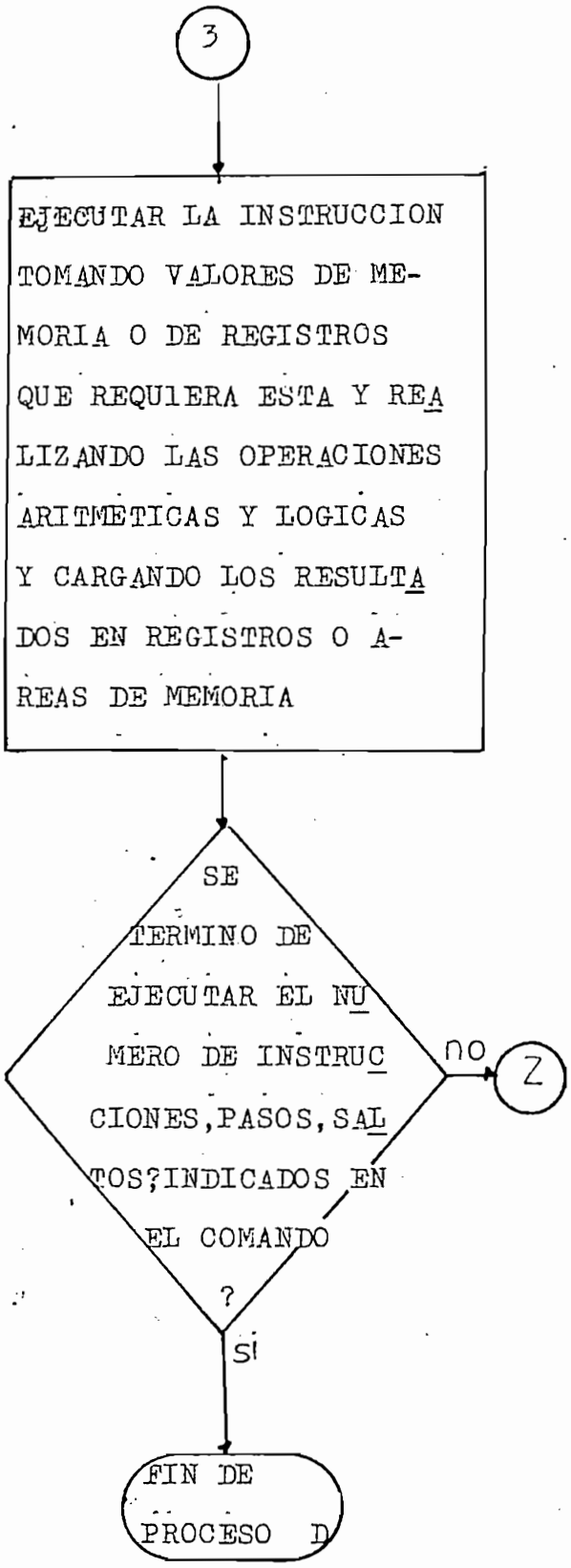
4-2-3 DIAGRAMA DE BLOQUES DEL SISTEMA SIMULADOR DEL M6800




INICIO PROCESO B
DECODIFICAR EL COMANDO DE ACUERDO AL # DE CODIGO DEL ARCHIVO CO2F (COMANDOS)
ES COMANDO DE CONTROL DE FORMATO O IMPRESION ?
si
GRAVAR EN EL ARCHIVO DE COMANDOS: PROPIEDADES PARA: BS, BE, SR.
FIN DE PROCESO B
no
ES COMANDO DE INICIALIZAR REGISTROS ?
si
PONER VALORES EN EL ARCHIVO DE REGISTROS SIMULADOS
FIN DE PROCESO B
no
ES COMANDO DE INICIALIZACION DE MEMORIA ?
si
PONER VALORES EN EL ARCHIVO DE MEMORIAS MEMRO
FIN DE PROCESO B
1


1
ES COMANDO DE CARGA DEL CODIGO OBJETO ?
si
TOMAR DEL ARCHIVO DEL PROGRAMA ENSAMBLADO EL CODIGO OBJETO Y PONERLO EN EL ARCHIVO DE MEMORIA MEMRO
FIN DE PROCESO B
no
ES COMANDO DE EJECUCION ?
no
IMPRIMIR ERROR: NO ES UN COMANDO DEFINIDO
FIN DE PROCESO B
si
REALIZAR EL PROCESO D
FIN DE PROCESO B


2
DECODIFICAR LA INSTRUCCION BUSCANDO EN EL ARCHIVO DE INSTRUCCIONES CO2P; OBTENIENDO UN CODIGO PARA LA EJECUCION; CODIGO DE DIRECCIONAMIENTO; TIPO DE REGISTRO, TIEMPO DE UP.
PONER EN EL CONTADOR DEL PROGRAMA LA DIRECCION DE LA SIGUIENTE INSTRUCCION A EJECUTARSE
AUMENTAR EL REGISTRO DE TIEMPO EN EL NUMERO DE CICLOS PARA CADA INSTRUCCION LEIDOS DEL ARCHIVO CO2P
3


3
EJECUTAR LA INSTRUCCION TOMANDO VALORES DE MEMORIA O DE REGISTROS QUE REQUIERA ESTA Y REALIZANDO LAS OPERACIONES ARITMETICAS Y LOGICAS Y CARGANDO LOS RESULTADOS EN REGISTROS O AREAS DE MEMORIA
SE TERMINO DE EJECUTAR EL NUMERO DE INSTRUCCIONES,PASOS,SALTOS?INDICADOS EN EL COMANDO ?
no
2
si
FIN DE PROCESO D

4-3 DESCRIPCION DE PROGRAMAS Y SUBRUTINAS.-PARAMETROS UTILIZADOS

Los programas y subrutinas se han desarrollado siguiendo los bloques explicados en el punto 4-2 y en lenguaje Fortran, cuya utilización ya ha sido justificada previamente.

4-3-1 PROGRAMAS IMPLEMENTADOS

PROGRAMA S6800

Objetivo: Este programa realizará el control del Simulador mediante comandos, macrocomandos y comandos a la tabla de macrocomandos.

Subprogramas llamados: COMAC, SIM, MAC, RECOM, SIMAC, EJEMA, ESMA.

Forma de utilización:

S6800
FILE 1: MEMRO
FILE 2: CO2P
FILE 3: REGIS
FILE 4: (Nombre del programa ensamblado)
FILE 5: COM
FILE 6: MACRO
FILE 7: ERR
FILE 8: NMONI

Funciones: Al controlar este programa el funcionamiento del Simulador permite el ingreso de los comandos descritos en el punto 3-4: reconociendo comandos, ejecutándolos, borrando o listando macrocomandos ; realizándose esto en los subprogramas llamados.

Diagrama de Flujo: Ver fig. 4-4-1

Funciones: Al controlar este programa el funcionamiento del Simulador permite el ingreso de los comandos descritos en el punto 3-4:reconociendo comandos,ejecutándolos,borrando o listando macrocomandos ; realizándose esto en los subprogramas llamados.

Diagrama de Flujo: Ver fig. 4-4-1

Listado: Ver Apéndice A1

PROGRAMA INT4

Objetivo: Cargar el archivo de instrucciones

Subprogramas llamados: HEDE4

Forma de utilización:

INT4
FILE 4: CO2P

Parámetros Utilizados:

INT(4)
NNEX(2) En 2 variables se guarda el operador en hexadecimal.

J Número de registro del archivo CO2P.

NPRO Código asignado a las instrucciones.

NDIR Número asignado a las instrucciones de acuerdo al tipo de direccionamiento

NREG Número para distinguir los tipos de registros en una instrucción.

NTIME Tiempo en ciclos de máquina para cada instrucción.

MOD Operador en decimal.

M Número de caracteres de una variable a ser transformados de hexadecimal a decimal.

Funciones: Guarda en el archivo CO2P:las instrucciones del M6800,código de cada instrucción,código para el tipo de direccionamiento,código para el tipo de registro,tiempo de la instrucción;además transforma las instrucciones de hexadecimal a decimal y las guarda así en el archivo.

Diagrama de flujo: Ver fig. 4-4-2

Listado: Ver Apéndice A2

PROGRAMA LECOD

Objetivo: Lectura del archivo de instrucciones.

Subprogramas llamados: DEC4

Forma de utilización:

LECOD

FILE 2: CO2P

J : Se asigna 3 dígitos decimales bajo las guías XXX

N : Se ingresa 3 dígitos decimales bajo las guías YYY

Parámetros utilizados:

NNHEX(2) Dos caracteres de los códigos de instrucción.

J Número donde se inicia la lectura del archivo.

N Número hasta el cual se leee el archivo.

NOD Código de instrucción en decimal.

NPRO Código de instrucción asignado para ejecución.

NDIR Número de direccionamiento

NREG Número de registro

PROGRAMA COMAN

Objetivo : Carga del archivo de comandos

Forma de utilización:

COMAN

FILE 5 :COM

J : Se ingresa 2 dígitos decimales bajo las guías XX

KCOM : Se ingresa bajo CC

NSEC : Dos dígitos bajo NN

JESP :Nueve dígitos bajo JJJJJJJJJ

M : Dos dígitos bajo MM

K : Dos dígitos bajo KK

Parámetros utilizados:

KCOM(2) Caracteres alfanuméricos del comando

J Número de registro del archivo de comandos donde se va a gravar.

NSEC Código de comando asignado para selección de comandos.

M Propiedad 1 de los comandos.

K Propiedad 2 de los comandos.

Funciones :Grabar en el archivo de comandos,los comandos u-

tilizados y propiedades de estos.Estas propiedades posteriormente,en la ejecución de comandos ,pueden ser cambiadas;por ejemplo si se necesita otra base de salida o de entrada,seleccionar otros registros,etc.

Diagrama de Flujo : Ver fig. 4-4-4

Listado : Ver Apéndice A4

PROGRAMA LCOM

Objetivo : Lectura del archivo de comandos.

Forma de utilización :

LCOM

FILE 5 :COM

J Dos dígitos bajo las guías XX

N Dos dígitos bajo las guías YY

Terminar el ingreso de valores con /*

Parámetros utilizados :

LCOM(2) Caracteres alfanuméricos de un comando

LSEC Código para selección de comandos

LESP Especificación en un comando

M,K Propiedades de los comandos

J Número donde se inicia la lectura de un archivo

N Número hasta donde se lee el archivo

Funciones :Lee los comandos almacenados y las características de los mismos.

Diagrama de Flujo : Ver fig. 4-4-5

Listado : Ver Apéndice A5

PROGRAMA CEROR

Objetivo :Grabación del archivo de errores

Forma de utilización :

CEROR

FILE 7 :ERR

NN Dos dígitos bajo las guías NN

INI,IFI Ingresar bajo NNFF

Terminar de ingresar datos con /*

Parámetros utilizados:

NN Número de registro del archivo

INI Número de registro donde empieza el enunciado de error

IFI Número de registro donde termina el enunciado de error

LER(80) Enunciado de un error.

Funciones : Graba en el archivo de errores ,posiciones de registros entre los cuales hay un error y los enunciados.

Diagrama de Flujo:Ver fig.4-4-6

Listado: Ver Apéndice A6

PROGRAMA LEROS

Objetivo : Lectura de errores

Subprogramas llamados : LERR

Forma de utilización :

LEROS

FILE 7 :ERR

J,N Ingresar bajo las guías XXYY

Terminar de ingresar J y N con /*

Parámetros utilizados :

J Número de registro donde se inicia la lectura del archivo.

N Número hasta donde se lee el archivo.

K Parámetro de lazo de lectura entre J y N

Funciones :Lee los errores guardados en el archivo ERR

Diagrama de Flujo : Ver fig. 4-4-7

Listado : Ver Apéndice A7

4-3-2 SUBRUTINAS LLAMADAS POR LOS PROGRAMAS ANTERIORES

SUBRUTINA COMAC

Definición : SUBROUTINE COMAC (LL,NR)

Objetivo :Ejecutar comandos a tabla de macrocomandos.

Subprogramas llamados : MAC,RECOM

Forma de utilización : CALL COMAC (LL,NR)

Parámetros utilizados:

LL(72) 72 caracteres alfanuméricos que pueden contener comandos,macrocomandos o comandos a tabla de macrocomandos.

NR < 0 Indica que no ha sido comando a macro
NR < 1 Indica que si ha sido comando a tabla de macrocomandos.

LS(72) Macrocomando leído del archivo.

NCOM(20) Guarda las posiciones de las comas en el comando BM; si no hay comas las variables de NCOM valen 0.

ICO Contador para caracteres:comas.

NAM(5) Aquí se almacena el nombre de un macro a borrar.

NM,NL,NB Definen las letras que forman un comando.

LIM Número de registro hasta el cual hay gravado un macro.

NBL Posición de un caracter blanco.

MI Posición donde se inicia el nombre de un macro a borrar.

N Posición donde termina el nombre de un macro a borrar.

MN Subíndice del vector NAM

NE
- 0 Hay un error en el nombre de un macro a borrar.
- 1 No hay error

IM1 Número de caracteres del nombre.

N1
- 0 No existe el macro en la tabla.
- 1 Sí existe el macro en la tabla.

K Registro en archivo de macros donde esta el que se pide borrar.

JJ,J2 Posición antes y después de desplazar un macro.

Datos de entrada : LL(72)

Datos de salida : NR

Funciones :Esta subrutina es parte del control del Simulador mediante comandos a la librería de macros,como son: LM(Listar macros) y BM(Borrar macros).

Diàgrama de Flujo : Ver fig. 4-4-8

Listado : Ver Apéndice A8

SUBRUTINA SIM

Definición : SUBROUTINE SIM(LL,IM,NSEC,JCO,M,K)

Objetivo : Realizar cada comando

Subprogramas llamados : ESPE,DEK,RENUM,DEBAS,READ,DECE4,DENT, IMPM,GRAT,EJET,WRITO,CONT,ENRE,LERR, ULTID.

Forma de Utilización : CALL SIM(LL,IM,NSEC,JCO,M,K)

Parámetros utilizados :

LL(72) 72 caracteres que contienen un comando.

IM Registro en el archivo de comandos donde está el que se va a ejecutar.

NSEC Código del comando leído del archivo de comandos.

JCO Especificación de comandos,por ejemplo 16 para BE.

M,K Propiedades de los comandos BE y BS.

NSED,NSEU Número de decenas y unidades en el código NSEC.

JESP Valor decimal obtenido de caracteres alfanuméricos.

NCAR Caracter alfabético que representa un registro simulado.

NUM Código de registros simulados del M6800

MPD Contador del programa simulado (en decimal).

MOD Código de operador simulado (en decimal).

MDI Dirección de la instrucción que se ejecuta.

IEA Dirección efectiva del operando en el programa.

NXD Registro índice(en decimal)

NAD Registro acumulador A (en decimal).

NBD Registro acumulador B(en decimal).

MCCRD Registro de condiciones (en decimal).

MSD Registro 'satck pointer'(en decimal).

MUL Ultima instrucción ejecutada(en decimal).

KCOM1,KCOM2 Caracteres que forman el nombre de un comando.

MM Número de dígitos hexadecimales transformados de un decimal.

N Número de dígitos hexadecimales para valores a memoria.

JBAS Valor de la base:2,8,10,16

NDID Dirección de memoria donde se empieza a poner valores, a ver valores o a ejecutar el programa.

NOO(18) Valores ingresados por los comandos:PM,VM,EJ,RN,RS,EI.

NERR
- 0 No hay error en comando y se pasa a ejecutarlo.
- 1 Hay error en comando ingresado.

NRD Número de valores a ser cambiados a decimal.

M4 Posición de comas.

NDD Número de localidades de memoria a visualizarse.

JDE Intervalo de líneas para dar encabezamiento.

NAU1,NAU2 Nombres auxiliares de comandos.

JBPM
- 0 Indica que no se ha utilizado comando PM
- 1 Indica que ya se ha utilizado el comando PM.

NER Número de error del que se pide un mensaje.

Datos de entrada : LL(72),IM,NSEC,JCO,M,K

Funciones:Realiza el control del Simulador mediante comandos.

Diagrama de Flujo:Ver fig.4-4-9 Listado:Apéndice A9

SUBRUTINA MAC

Definición : SUBROUTINE MAC(L5,NE,I1)

Objetivo : Verificar la formación del nombre de un macro.

Subprogramas llamados : EXPLO

Forma de utilización : CALL MAC(L5,NE,I1)

Parámetros utilizados :

L5(5) Nombre de un macrocomando.

LE(5) Arreglo auxiliar para el nombre de un macro.

NE
- 0 Hay error en el nombre.
- 1 No hay error en el nombre.

M(4,5) Matriz de transición de estados para reconocer un nombre

I1 Posición del siguiente caracter luego del nombre.

NT Tipo de caracter explorado por la subrutina EXPLO, NT=1:Letra;NT=2:Dígito;NT=3:Blanco;NT=4:Signo.

II Estado de la matriz de transición.

Datos de entrada : L5(5)

Datos de salida : NE,I1

Funciones : Esta subrutina analiza que el nombre de un macro-comando este bien formado,es decir,que el primer caracter sea una letra,los siguientes pueden ser letras o números y que tenga máximo cuatro caracteres sin haber la posibilidad de contener signos.

Diagrama de Flujo : Ver fig. 4-4-10

Listado : Ver Apéndice A10

SUBRUTINA RECOM

Definición : SUBROUTINE RECOM (IM1,L5,N1,K)

Objetivo : Subrutina que ve en la librería de macrocomandos si existe definido un macro.

Forma de utilización : CALL RECOM (IM1,L5,N1,K)

Parámetros utilizados :

IM1 Número de caracteres del nombre.

L5(5) Nombre del macrocomando.

N1
- 1 Indica que si hay el macrocomando en la tabla.
- 0 Indica que no hay el macrocomando en la tabla.

K Registro en el archivo de macros donde se reconoce uno.

KDCOM Número de macrocomandos almacenados menos uno.

NAM(5) Nombre de un macro leído del archivo para comparación.

IM2 Posición luego del nombre.

Datos de entrada : IM1,L5(5)

Datos de salida : N1,K

Funciones : Busca en la librería de macrocomandos si hay definido un macro.Cuando el programa S6800 llama a esta subrutina y si esta indica que si hay definido un macro;asume que lo que ha pedido el usuario al poner el nombre de un macro es ejecutarlo ; mientras si no está gravado el macro, se procede a hacerlo.

También se llama a esta subrutina cuando se desea borrar macros;siendo esta la que reconoce si hay el macro que se pide borrar.

Diagrama de Flujo : Ver fig. 4-4-11

Listado : Ver Apéndice A11

SUBRUTINA SIMAC

Definición: SUBROUTINE SIMAC (LL,NEM,NPOC,NPUN)

Objetivo : Comprobar la sintaxis del macrocomando:paréntesis, comas,puntos,etc.

Forma de utilización :CALL SIMAC(LL,NEM,NPOC,NPUN)

Parámetros utilizados :

LL(72) Macrocomando con su nombre y contenido.

NEM
- 0 El macrocomando es sintácticamente correcto.
- 1 Hay error en la definición del macrocomando

NPOC(2) Posiciones de los paréntesis del macrocomando.

NPUN(10) Posiciones de puntos de separación entre comandos.

LE(60) Arreglo auxiliar de LL

NBLC(10) Posiciones de los caracteres blancos.

CORE Signo de abrir paréntesis.

COSE Signo de cerrar paréntesis.

I1,I2 Contadores de abrir y cerrar paréntesis.

LP Contador del número de puntos en un macrocomando.

I3 Contador de caracteres blancos.

Datos de entrada : LL

Datos de salida : NEM,NPOC,NPUN

Funciones : Identifica las posiciones de puntos de separación entre comandos,posiciones de signos de abrir y cerrar paréntesis en la definición de macrocomandos ; además comprueba que estén balanceados los paréntesis y que no haya espacios en blanco luego de puntos.Es decir,esta subrutina se encarga de la sintaxis del macrocomando.

Diagrama de Flujo : Ver fig. 4-4-12

Listado : Ver Apéndice A12

SUBRUTINA EJEMA

Definición : SUBROUTINE EJEMA (K,MER)

Objetivo : Ejecuta el macrocomando, realizando comando por comando

Subprogramas llamados : SIM

Forma de utilización : CALL EJEMA (K,MER)

Parámetros utilizados :

K Número de registro en el archivo de macros.

MER
- 0 No hay error y se han ejecutado los comandos.
- 1 Hay error en el macro y no se ejecuta.

LS(60) Macro que se va a ejecutar.

NPUN(10) Posiciones de puntos entre comandos del macro.

NPOC(20) Posiciones de paréntesis en el macro.

LR(72) Arreglo que toma cada comando.

KCOM(2) Caracteres del nombre de un comando

I1,I2 Posiciones donde empieza y termina un comando.

IM Puntero del archivo de comandos.

NSEC Código de un comando.

JCO,MN,KL Propiedades de los comandos

Datos de entrada : K

Datos de salida : MER

Funciones : En base a la posición K de un macrocomando en la librería de macros, esta subrutina ejecuta los comandos grabados en este.

Se realiza la ejecución comando por comando. Además identifica errores sea por mala formación del macro grabado o porque un comando no esta entre los implementados para el Simulador.

Diagrama de Flujo : Ver fig. 4-4-13

Listado : Ver Apéndice A13

SUBRUTINA HEDE4

Definición : SUBROUTINE HEDE4 (MOP,MOD,M)

Objetivo : Cambiar un número en base hexadecimal a decimal.

Parámetros utilizados :

MOP(4) Número de hasta cuatro caracteres hexadecimales.

MOD Número en base decimal.

M Variable que indica cuantos caracteres hexadecimales contiene el número que se va a transformar.

NAUX(4) Equivalente decimal de cada caracter hexadecimal.

Datos de entrada : MOP,M

Datos de salida : MOD

Funciones :Cambiar los operandos de hexadecimal a decimal,porque las operaciones que se simularán se harán en decimal.

Diagrama de Flujo : Ver fig. 4-4-15

Listado : Ver Apéndice A15

SUBRUTINA ESMA

Definición : SUBROUTINE ESMA (LL,NPUN,NPOC,MLL)

Objetivo : Escribir un macrocomando en la librería de macros.

Forma de utilización : CALL ESMA (LL,NPUN,NPOC,MLL)

Parámetros utilizados :

LL(72) Macrocomando a ser guardado en la tabla de macros.

NPUN(10) Posiciones depuntos en un macro.

NPOC(2) Posiciones de paréntesis en macro.

MLL
- 1 Librería de macros llena.
- 0 Librería de macros no esta llena.

LE(60) Arreglo auxiliar de LL

LIM Número demmacros grabados en la tabla de macros.

Datos de entrada :LL,NPUN,NPOC

Datos de salida : MLL

Funciones:Gravación de un macro y posiciones de sus puntos,part.

Diagrama de Flujo :Ver fig. 4-4-14 Listadp : Apéndice A14

SUBRUTINA DEC4

Definición : SUBROUTINE DEC4(INDE,NHEX,NBCD)

Objetivo : Cambiar un número decimal a hexadecimal de dos caracteres.

Forma de utilización: CALL DEC4 (INDE,NHEX,NBCD)

Parámetros utilizados :

INDE Número en base decimal

NHEX(2) Caracteres del número en base hexadecimal.

NBCD(2) Equivalente decimal de los caracteres hexadecimales.

NUMX(16) 16 caracteres hexadecimales.

Datos de entrada : INDE

Datos de salida :NHEX,NBCD

Funciones : Esta subrutina se utiliza para cambiar un número en base decimal a hexadecimal de dos caracteres;esto se utiliza para presentación de registros en base hexadecimal.

Diagrama de Flujo : Ver fig. 4-4-16

Listado : Ver Apéndice A16

SUBRUTINA LERR

Definición :SUBROUTINE LERR (NER)

Objetivo : Lee un error del archivo del archivo de errores.

Forma de utilización : CALL LERR (NER)

Parámetros utilizados :

NER Número de error que se desea saber el significado.

LER(80) Mensaje de error.

NER1 Posición en el archivo de errores donde están las direcciones de registros donde inicia y termina el mensaje de un error.

INI,IFI Registros donde inicia y termina el enunciado de error

Funciones : En base al número asignado a un error ,se imprime un mensaje ampliando su significado.

Diagrama de Flujo : Ver fig. 4-4-17

Listado : Ver Apéndice A17

4-3-3 SUBPROGRAMAS LLAMADOS POR LAS SUBRUTINAS DEL PUNTO 4-3-2

SUBRUTINA ESPE

Definición :SUBROUTINE ESPE (N1,N2,JES)

Objetivo : Transformar dos caracteres alfanuméricos a su equivalente numérico en decimal.

Subprogramas llamados : CARD

Forma de utilización : CALL ESPE(N1,N2,JES)

Parámetros utilizados :

N1,N2 1ro. y 2do. caracteres alfanuméricos.

JES Valor decimal.

NCAR1,NCAR2 Caracteres auxiliares para N1 y N2

NUM1,NUM2 Equivalentes numéricos de N1 y N2

Datos de entrada: N1,N2

Datos de salida :JES

Funciones :Transforma los caracteres leídos como alfanuméricos en los comandos:BE,BS,CO a su número equivalente.

Diagrama de Flujo :Ver fig. 4-4-18 Listado Ver Apéndice A18

SUBRUTINA DEK

Definición : SUBROUTINE DEK (JES,M,K)

Objetivo : Asignar determinados códigos de acuerdo a la base.

Forma de utilización : CALL DEK(JES,M,K)

Parámetros utilizados :

JES Valor de la base (en decimal)

M Número de caracteres que tendrían las variables de dos bytes en las diferentes bases.

K Código asignado a las bases: K=1 binaria
K=2 octal
K=3 hexadecimal
K=4 decimal

Datos de entrada : JES

Datos de salida : M,K

Funciones :Los códigos de las diferentes bases se utilizan en distintos procesos para cada una de ellas.

Diagrama de Flujo : Ver fig. 4-4-19

Listado :Ver Apéndice A19

SUBRUTINA RENUM

Definición : SUBROUTINE RENUM (NC,NUM)

Objetivo :Asignar un número a cada registro simulado.

Forma de utilización : CALL RENUM(NC,NUM)

Parámetros utilizados :

NC Caracter alfabético que representa un registro.

NENT(9) Caracteres alfabéticos iniciales de los registros.

NUM Número asignado a los registros:P-1 ; O-2 ; I-3 ; E-4 ; X-5 ;A-6 ; B-7 ; C-8 ; S-9

Datos de entrada : NC

Datos de salida : NUM

Funciones : Con esta subrutina se obtiene un código numérico para cada registro;con lo que se puede guardar información de los registros seleccionados en forma de un número.

Diagrama de Flujo : Ver fig. 4-4-20

Listado : Ver Apéndice A20

SUBRUTINA DEBAS

Definición : SUBROUTINE DEBAS

Objetivo : Imprimir los registros seleccionados en la base de salida seleccionada.

Subprogramas llamados : CO210

Forma de utilización : CALL DEBAS

Parámetros utilizados:

NC16(9,16) 9 registros simulados con los caracteres que habría en dos bytes para cualquier base.

NLEC(9) Registros leídos del archivo REGIS.

NENT(9) Letras iniciales de cada registro.

NAU1,NAU2 Nombre de un comando en dos caracteres.

NAU3 Código del comando:Base de Salida

JBAS Base de salida.

M Número de caracteres de un registro en base JBAS

N Número de registros a ser transformados de decimal a base JBAS.

N1,N2 Caracteres del comando SR (Seleccione Registros)

N3 Código del comando SR.

JESP Indica los registros seleccionados por el comando SR

NK Variable auxiliar para M

J Exponente para ir decomponiendo el número JESP en sus dígitos que representan los registros.

NCAR Número que representa el registro seleccionado.

Funciones : Por medio de esta subrutina se visualiza los registros seleccionados con el comando SR en la base de salida seleccionada por el comando BS.

Diagrama de Flujo : Ver fig. 4-4-21

Listado : Ver Apéndice : A21

SUBRUTINA REAO

Definición : SUBROUTINE REAO(MPD,MOD)

Objetivo : Lectura del archivo de memoria simulada

Forma de utilización : CALL REAO(MPD,MOD)

Parámetros utilizados :

MPD Puntero del archivo de memoria simulada MEMRO ,de donde se lee un valor.

MOD Valor leído del archivo MEMRO

Datos de entrada : MPD

Datos de salida : MOD

Funciones :Se implementa esta subrutina para realizar la lectura del archivo de memoria simulada;y debido a que en un archivo no existe el registro 0 ,se asigna un registro luego del último .

Diagrama de Flujo :Ver fig, 4-4-22

Listado : Ver Apéndice A22

SUBRUTINA DECE4

Definición : SUBROUTINE DECE4(INDE,NHEX,L)

Objetivo : Cambiar un número decimal a hexadecimal de 4 caracteres.

Forma de utilización : CALL DECE4(INDE,NHEX,L)

Parámetros utilizados:

INDE Número en base decimal.

NHEX(4) Cuatro caracteres del número en base hexadecimal.

L Número de caracteres hexadecimales a obtenerse.

NAUX(4) Equivalentes decimales de caracteres hexadecimales.

NUMX(16) 16 caracteres posibles en base hexadecimal.

M Valor al cual se acumula un nuevo caracter hexadecimal.

Datos de entrada : INDE,L

Datos de salida : NHEX

Funciones :Utilizase esta subrutina para visualizar valores en base hexadecimal.

Diagrama de Flujo : Ver fig. 4-4-23 Listado :Apéndice A23

SUBRUTINA DENT

Definición : SUBROUTINE DENT (NDID,NOO,M,NERR,LL,N,JBAS,NRD,M4)

Subprogramas llamados : CB16

Forma de utilización :

CALL DENT(NDID,NOO,M,NERR,LL,N,JBAS,NRD,M4)

Parámetros utilizados :

NDID Localidad de memoria donde se inicia la grabación o lectura de datos ,o la ejecución de un programa.

NOO(18) Datos a ser leídos o gravados o número de pasos a ejecutar.

M Caracteres que hay en dos bytes en cualquier base.

NERR

1 Hay un error en comando ingresado.

0 No hay error en valores ingresados.

LL(72) Arreglo que contiene uno de los comandos:PM,VM,EJ, RN,EI,RS.

N Número de caracteres que tienen los valores en una base de entrada.

JBAS Base de entrada.

NRD Número de valores a ser cambiados a decimal.

M4 Posiciónes de comas al ingresar valores.

KD16(16) Valor en la base de entrada,de la primera posición de memoria donde se inicia,lo indicado por:PM,VM,EJ,RN, EI,RS.

KD8(16) Número de memorias a leerse,o número de pasos a ejecutarse o un valor a memoria.

M1,N1 Variables auxiliares para M y N.

NDD Valor en decimal de un dato ingresado en cualquier base

Datos de entrada : M,LL,N,JBAS,NRD,M4

Datos de salida : NDID,NOO,NERR

Funciones : Se utiliza esta subrutina en comandos,para poner valores en localidades de memoria,leer un número de valores de memoria,especificar donde se empieza a ejecutar un programa y cuantos pasos hay que ejecutarse.

Además si se desea continuar poniendo valores en memoria a partir de la última localidad ingresada se utiliza una entrada a esta subrutina en la sentencia ENTRY CONT.

Diagrama de Flujo : Ver fig. 4-4-24

Listado : Ver Apéndice A24

SUBRUTINA IMPM

Definición : SUBROUTINE IMPM (NDID,NDO)

Objetivo : Imprimir las direcciones y contenidos de memoria

Subprogramas llamados : CO210

Forma de utilización : CALL IMPM(NDID,NDO)

Parámetros utilizados :

NDID Dirección de memoria donde se empieza a ve valores

NDO Número de localidades de memoria a visualizarse.

ND(9) En la 1ra. variable se guarda la dirección de memoria.

KCOM1,KCOM2 Nombres auxiliares del comando BS.

NSEC Código del comando BS

JBAS Base de salida:2,8,10,16.

M Número de caracteres en cualquier base para números de dos bytes.

K Código para base de salida.

MN Ultima localidad donde se graba un valor.

JJ Puntero del archivo de memoria.

N Número de registros a ser cambiados a la base JBAS.

L Número de caracteres en la base JBAS para una palabra de memoria.

NC16(9,16) Caracteres de una dirección de memoria en base JBAS

NK16(9,16) Caracteres de una palabra de memoria.

Datos de entrada : NDID,NDO

Funciones : Dentro de los comandos que controlan el Simulador,para visualizar memoria tenemos el comando VM ;este utiliza la subrutina IMPM ,permitiendo ver desde una localidad de memoria ,un número hasta de 256 palabras de memoria.

Diagrama de Flujo : Ver fig. 4-4-25

Listado : Ver Apéndice A25

SUBRUTINA GRAT

Definición : SUBROUTINE GRAT

Objetivo : Transferir el código objeto de un programa ensamblado al archivo de memoria simulada.

Subprogramas llamados : HEDE4,WRITO

Forma de utilización : CALL GRAT

Parámetros :

L(26) Arreglo de los caracteres alfabéticos.

LD(4) Dirección de memoria donde se empiezan a guardar valores de una línea de programa.

LOP(4) En dos caracteres se guarda el código de un operador.

LR1(4) Primer operando ,si lo hay,en una instrucción.

LR2(4) 2do. operando para instrucciones de 3 bytes.

M(70) Una línea de un programa ensamblado.

AST Símbolo asterisco (*)

BLANCO Espacio en blanco.

NOP Operador en decimal

LOR1,LOR2 1ro. y 2do. operandos de una instrucción(en decimal)

NBO
- 1 Indica que no se encuentra la etiqueta ORG
- 0 Ya se encontró la etiqueta ORG

MD,N Número de caracteres en una palabra en hexadecimal que se desea pasar a decimal.

N1 Número de registros en el archivo de programa ensamblado.

N2 Número de líneas de programa ensamblado.

J Posición de un registro en el archivo de programa ensamblado.

Funciones : Esta subrutina encadena el Cross Assembler y el Simulador;obteniendo del programa ensamblado el código objeto y guardando en la memoria simulada.

Diagrama de Flujo : Ver fig. 4-4-26

Listado : Ver Apéndice A26

SUBRUTINA EJET

Definición : SUBROUTINE EJET (NDID,NREP,JESP)

Objetivo : Decodificar y ejecutar las instrucciones.

Subprogramas llamados : REAO,COND4,CO210,NOMI,REDOS,NUML4,
ME254,HAC4,CR4,BORR4,NE4,CER4,SOBR4,
CONB4,ARRE4,LOGN4,WRITO,SOB4,CAR4,DEC4,
RO4,BALD4,REMOS,GUARD.

Forma de utilización : CALL EJET (NDID,NREP,JESP)

Parámetros utilizados :

NDID Dirección de memoria donde se inicia la ejecución .

NREP De acuerdo al valor de JESP,la variable NREP representa:

JESP=1 NREP:Número de pasos de programa a ejecutarse

JESP=2 NREP:Número de pasos de programa a rastrear.

JESP=3 NREP:Número de saltos de programa a rastrear.

JESP=4 NREP:Número de instrucciones a ejecutar.

LA8(8) Arreglo lógico para los 8 bits del acumulador A así:

bit1=1 LA8(1)=TRUE

bit1=0 LA8(1)=FALSE

LB8(8) Arreglo lógico para los 8 bits del acumulador B

LM8(8) Arreglo lógico para los 8 bits de una palabra de memoria

LR8(8) Arreglo lógico, resultado de alguna operación lógica de los arreglos anteriores.

LC8(8) Arreglo lógico para los bits del registro de condiciones

LL Resultado de una operación lógica previa a obtener el valor de la bandera V (se basa en la nota 6 del set)

AXL Variable lógica=TRUE, utilizada para probar expresiones

NNOV Resultado de la operación N ⊙V , siendo N y V banderas.

NC8(8) Arreglo lógico en base al registro de condiciones.

NA8(8) 8 bits del acumulador A.

NB8(8) 8 bits del acumulador B.

NCCR(8) En 6 variables se guardan los bits de banderas.

NLEC(9) Registros simulados del microprocesador M6800 (en decimal)

NC16(9,16) Caracteres hexadecimales de NLEC

NV8(8) 8 bits del byte más significativo del registro índice.

ITT	Puntero del archivo de instrucciones.
I8	Puntero del archivo de códigos nmónicos.
MODO	Código de operador (en decimal)
IEAD	Dirección efectiva del operando.
JK	Cuenta del número de pasos de programa que se ejecutan
NH	Valor de la bandera HALF-CARRY
NC	Valor de la bandera CARRY
NN	Valor de la bandera NEGATIVE
NV	Valor de la bandera OVERFLOW
NZ	Valor de la bandera ZERO
M4	Número de caracteres hexadecimales para una palabra de dos bytes.
JBAS	Base a la cual se cambian los valores de los registros
N9	Número de registros a ser cambiados de base.
MIN	Instrucción leída en la dirección MPD, en la memoria.
NPRO	Código de la instrucción.

MCCRD Registro de condiciones (en decimal) .

MSD 'Stack Pointer ' simulado (en decimal) .

NTIM Registro que acumula el tiempo en ciclos de máquina que ocupa cada instrucción ejecutada.

KAX1,KAX2 Nombre del comando CO

NSAX Código del comando CO

NESP 0 Se escoge el código objeto para salida de un rastreo.

1 Se escoge código nmónico para salida de un rastreo.

KCOM1,KCOM2 Nombre del comando DE

JDE Intervalo de líneas de programa simulado.al cabo de las cuales se presenta el rotulado de registros.

JK5 Cuenta del número de líneas ejecutadas.

LBRAN Contador del número de saltos de programa realizados.

JB 0 Indica que no hay un salto en el programa simulado.

1 Indica que hubo un salto en el programa simulado.

NUMBY Número de bytes de una instrucción.

NM8(8) 8 bits de una palabra de memoria.

NBCD(2) Equivalente decimal de cada caracter hexadecimal.

INT(198,5) Arreglo al que se ha volcado el archivo de instrucciones CO2P

NMON(4) Código nmónico de una instrucción.

MNO(108,16) Arreglo que contiene los códigos nmónicos de todas las instrucciones,copiado del archivo NMONI

NHEX(2) Dos caracteres hexadecimales del acumulador A

LR,LS,LI,LN,LO Nombres de las letras R,S,I,N,O respectivamente.

MPD Registro simulado del contador del programa (en decimal)

MOOD Código de operador (en decimal)

MDI Dirección de la instrucción (en decimal).

IEA Dirección efectiva del operando.

NXD Registro índice simulado (en decimal).

NAD Acumulador A simulado(en decimal).

NBD Acumulador B simulado (en decimal)

LDR Número asignado al tipo de direccionamiento:1-Inmediato;2-Directo;3-Indexado;4-Extendido;5-Inherente; 6-Relativo.

NREG Tipo de Registro;por ejemplo para la instrucción ADD se tiene : 1-Registro A;2-B y 3-A,B .

NT Número de ciclos que toma una instrucción.

LAX 2do.byte en una instrucción de dos bytes o 3ro. en una de tres bytes.

LAX1 2do. byte en una instrucción de 3 bytes.

MDD Número de decenas en el código de la instrucción NPRO

MUD Número de unidades en el código de la instrucción.

NCX Valor del'carry' para suma o resta con carry.

NBX Variable auxiliar para contener el operando o el acumulador B.

MOD Operando endecimal.

NAX Contiene el acumulador A o el B.

MCD Variable auxiliar para el resultado de una operación de suma.

NRD Resultado de una operación que sirve para analizar el cambio de banderas de condición.

KBX,KCX Variables auxiliares que sirven para analizar el cambio de la bandera 'CARRY' (para la resta 'BORROW')

K1CRD Valor de la bandera 'CARRY'.

MM Variable igual a cero.

KCCRD Valor de la bandera 'CARRY'.

MEM Complemento de dos o de uno ,de un valor de memoria.

NN1 Valor a incrementar a los acumuladores o a una palabra de memoria.

NN2 Valor de prueba para ver si cambia la bandera V.

NVD Guarda el valor de acumuladores o memorias para comparar con NN2 (en base a las notas 4 y 5 del set).

NCX Variable auxiliar para guardar la bandera de 'CARRY'.

MOD1 3er.byte en instrucciones de 3 bytes.

NCOM Valor para obtener el complemento de un número de dos bytes.

NOV Byte superior de un operando de 2 bytes (en decimal).

NC2 Byte menos significativo de un operando de dos bytes.

NXX Byte menos significativo del registro índice.

NRR Resta de bytes menos significativos, cumpliendo la nota 8 del set de instrucciones.

MOS Incremento o diminución en 1 a registros de 2 bytes.

NRH,NRL Byte más y menos significativo de registros de 2 bytes.

MODC Número de líneas de salto en un programa simulado.

MODCC Complemento de dos ,para el número de líneas de salto.

MPDSH,MPDSL Byte más y menos significativo del contador de programa simulado.

MEM1,MEM2 Dirección de memoria donde está la localidad de inicio del programa de interrupción.

KL3,KL4 Datos para ver si se desea interrupción.

KL1,KL2 Nombre de comando de interrupción.

Datos de entrada : NDID,NREP,JESP

Funciones : Esta subrutina decodifica las instrucciones, obtiene información de la instrucción decodificada como es : código, tipo de direccionamiento, tipo de registro, tiempo de cada instrucción.

También,esta subrutina,ejecuta la instrucción decodificada;realizando esto el número de veces que se indica en los comandos así:número de pasos,saltos,instrucciones.Imprimiendo luego de cada ejecución o luego de cada salto oa al final del programa simulado.

Diagrama de Flujo : Ver fig. 4-4-27

Listado : Ver Apéndice A27

SUBRUTINA WRITO

Definición : SUBROUTINE WRITO(MPD,MOD)

Objetivo :Grabar en memoria simulada

Forma de utilización : CALL WRITO(MPD,MOD)

Parámetros utilizados :

MPD Dirección de memoria simulada donde se desea grabar

MOD Valor a guardar en la localidad MPD.

Datos de entrada : MPD,MOD

Funciones : Debido a que en un archivo no se puede guardar un dato en un registro 0,esta subrutina nos permite grabar en una localidad que representa la dirección 0 de memoria.

Diagrama de Flujo : Ver fig. 4-4-28

Listado :Ver Apéndice A28

SUBRUTINA ENRE

Definición : SUBROUTINE ENRE (LL,NERR)

Objetivo : Poner valores en registros.

Subprogramas llamados : CARD,RENUM,CB16

Forma de utilización : CALL ENRE (LL,NERR)

Parámetros utilizados :

LL(72) Caracteres del comando PR(Poner en registros).

NERR

0 No hay error en comando.

1 Hay un error en comando.

N11(9) Número al que se divide el número de caracteres hexadecimales en dos bytes para obtener el de cada registro

KD08(16) Caracteres de un valor a asignarse a un registro.

NC(9) Registros simulados.

N1C(9) Valores numéricos del arreglo N11

KCOM1,KCOM2 Nombre del comando BE.

NSEC Código del comando BE.

JBAS Base de entrada de datos.

M Número de caracteres en una palabra de dos bytes.

K Código para la base de entrada.

NOF2 Separación hasta donde hay un nuevo valor de registro

NS Incremento para NOF2

NRD Número de registros que han cambiado de valor

NCAR Letra inicial de un registro.

NUM Número asignado a cada registro.

N,LN Número de caracteres de un registro.

IN Posición donde está el valor de un registro.

NREP Valor decimal que se va a dar a un registro.

Función : Se utiliza esta subrutina en la inicialización de registros por medio del comando PR (Poner en Registros);sirviendo esto para dar condiciones iniciales a registros antes de ejecutar un programa.

Diagrama de Flujo : Ver fig. 4-4-29

Listado : Ver Apéndice A29

SUBRUTINA ULTID

Definición : SUBROUTINE ULTID

Objetivo : Impresión de la dirección efectiva del operando.

Subprogramas llamados : CO210

Forma de utilización : CALL ULTID

Parámetros utilizados :

NLEC(9) Registros simulados

NC16(9,16) Caracteres de cada registro en cualquier base.

MDI Dirección de la instrucción en un programa simulado.

MOD Operador endecimal

IEA Dirección efectiva del operando en un programa simulado

N Número de registros a transformarse a otra base.

NNX1,NNX2 Nombre del comando UD (Ultima Dirección).

NSEC Código del comando UD.

JBAS Base de salida para imprimir la última dirección

M Código para la base de salida.

Funciones : Se llamará a esta subrutina cuando se especifique el comando del Simulador que pide se imprima la última dirección del operando en la última instrucción ejecutada en un programa.

Daiagrama de flujo : Ver fig. 4-4-30

Listado : Ver Apéndice A30

SUBRUTINA EXPLO

Definición ; SUBROUTINE EXPLO (LEX,NT)

Objetivo : Explorar caracteres ingresados y darles un código

Forma de utilización : CALL EXPLO (LEX,NT)

Parámetros utilizados ;

LEX Caracter ingresado al cual se va a dar un código.

NT Código asignado al caracter LEX : NT=1 - Letra ;
NT=2 - Número ; NT=3 - Espacio en Blanco ;
NT=4 - Caracter especial.

LN(26) Caracteres alfabéticos.

LD(10) Caracteres numéricos.

Datos de entrada : LEX

Datos de salida : NT

Funciones : Se utiliza esta subrutina para reconocer la formación del nombre de un macrocomando; se la llamará para ver el tipo de caracteres que forman el nombre.

Diagrama de Flujo : Ver fig. 4-4-31

Listado : Ver Apéndice A31

4-3-4 SUBRUTINAS LLAMADAS EN LOS SUBPROGRAMAS DE 4-3-3

SUBRUTINA CARD

Definición : SUBROUTINE CARD (NCAR,NUM)

Objetivo : Cambiar un caracter alfanumérico a numérico.

Forma de utilización : CALL CARD (NCAR,NUM)

Parámetros utilizados :

NCAR Caracter alfanumérico de entrada .

NENT(10) Dígitos decimales.

NUM Caracter numérico correspondiente a NCAR.

Datos de entrada :NCAR

Datos de salida : NUM

Fumciones : Al llamar a esta subrutina,los datos leídos en un comandoen forma de caracteres alfanuméricos son cambiados a numéricos.

Diagrama de Flujo : Ver fig, 4-4-32

Listado : Ver Apéndice A32

SUBRUTINA CO210

Definición : SUBROUTINE CO210 (NL,JBAS,M,NC16,N)

Objetivo : Transformar variables en base decimal a otra base.

Subprogramas llamados : COC4

Forma de utilización : CALL CO210 (NL,JBAS,M,NC16,N)

Parámetros utilizados :

NL(9) Arreglo de hasta 9 variables,que pueden ser cambiadas a otra base.

JBAS Base a la cual se cambian los valores ingresados.

M Número de caracteres de una variable de 2 bytes en base JBAS.

NC16(9,16) Caracteres en base JBAS de las variables de NL.

NCON(16) Caracteres que forma una variable.

N Número de variables a ser cambiadas a otra base.

Datos de entrada : NL,JBAS,M,N

Datos de salida : NC16

Funciones : De los 9 registros utilizados en el Simulador, el número N de ellos,pueden ser cambiados por medio de esta subrutina,de base decimal a base binaria,octal o hexadecimal y ser presentados de esta forma con el comando VR(Ver Registros).

Diagrama de Flujo : Ver fig. 4-4-33

Listado : Ver Apéndice A33

SUBRUTINA CB16

Definición : SUBROUTINE CB16(KALF,NDID,M,JBAS)

Objetivo :Cambiar un número de cualquier base a base decimal.

Subprogramas llamados : CARD,HEDE4

Forma de utilización : CALL CB16(KALF,NDID,M,JBAS)

Parámetros utilizados:

KALF(16) Caracteres alfanuméricos de un número en base JBAS.

NDID Número en base decimal.

M Número de caracteres en variable de 2 bytes .

KD16(16) Caracteres numéricos correspondientes a KALF(16).

NDEX(4) Caracteres en base hexadecimal.

Datos de entrada : KALF,M,JBAS

Datos de salida : NDID

Funciones : Obtiene el equivalente en base decimal de un numero ingresado en cualquier base .

Diagrama de Flujo : Ver fig. 4-4-34 Listado : Apéndice A34.

SUBRUTINA COND4

Definición : SUBROUTINE COND4 (IDD,L)

Objetivo : Cambiar un número decimal a binario(en arreglo).

Forma de utilización : CALL COND4(IDD,L)

Parámetros utilizados :

IDD Número en base decimal.

L(8) Número en base binaria (en arreglo de 8 variables).

IK Mantiene el número que va descomponiéndose en bits.

Datos de entrada : IDD

Datos de salida : L

Funciones : Se utiliza esta subrutina para descomponer un numero decimal en bits;esto es necesario para simular operaciones con bits.

Diagrama de Flujo : Ver fig. 4-4-35

Listado : Ver Apéndice A35

SUBRUTINA NOMI

Definición : SUBROUTINE NOMI (MODO,MNO,NMON)

Objetivo : Obtener el código nmónico del operador.

Subprogramas llamados : HEDE4

Forma de utilización : CALL NOMI (MODO,MNO,NMON)

Parámetros utilizados :

MODO Código del operador.

MNO(108,16) Arreglo con los códigos nmónicos de todos los operadores del M6800 copiado del archivo NMONI.

NMON(4) Código nmónico del operador .

MOP(2) Código del operador en hexadecimal para comparación.

NMD Código del operador en decimal para comparación.

Datos de entrada : MODO,MNO

Datos de salida : NMON

Funciones :En base a esta subrutina se podrá imprimir el código nmónico de instrucciones ejecutadas.

Diagrama de Flujo : Ver fig. 4-4-36 Listado : Ver Apéndice A36

SUBRUTINA REDOS

Definición : SUBROUTINE REDOS (NRD)

Objetivo :Reduce un número mayor a 2 bytes a la parte correspondiente que hay en 2 bytes unicamente.

Forma de utilización : CALL REDOS (NRD)

Parámetros utilizados : NRD

NRD Número a ser reducido.

Datos de entrada : NRD

Datos de salida : NRD

Funciones : Se utiliza esta subrutina para reducir los valores que toman los registros simulados de dos bytes a su capacidad correspondiente.

Diagrama de Flujo : Ver fig. 4-4-37

Listado : Ver Apéndice A37

SUBRUTINA NUML4

Definición : SUBROUTINE NUML4 (NND,LL8)

Objetivo :Cambiar un número decimal a un arreglo lógico de bits

Subprogramas llamados : COND4

Forma de utilización : CALL NUML4(NND,LL8)

Parámetros utilizados :

NND Número decimal.

LL8(8) Arreglo de variables lógicas correspondientes a los bits del número NND.

N8(8) Arreglo de bits del número NND

Datos de entrada : NND

Datos de salida : LL8(8)

Funciones : Cambia un número a ... variables lógicas correspondientes a sus bits; siendo esto necesario para simular las operaciones lógicas del M6800.

Daiagrama de Flujo : Ver fig. 4-4-38

Listado : Ver Apéndice A38

SUBRUTINA ME254

Definición : SUBROUTINE ME254 (NN)

Objetivo : Reducir un número a su parte correspondiente menor que 256 en decimal.

Forma de utilización : CALL ME254 (NN)

Parámetros utilizados :

NN Número a ser reducido.

Datos de entrada : NN

Datos de salida : NN

Funciones : Se utiliza esta subrutina para reducir el contenido de registros de un byte a su capacidad correspondiente.

Diagrama de flujo : Ver fig. 4-4-39

Listado : Ver Apéndice A39

SUBRUTINA HAC4

Definición : SUBROUTINE HAC4 (NAX,NBX,NCX,NH)

Objetivó: Obtención del valor de la bandera H ('HALF CARRY')

Subprogramas llamados : NH4

Forma de utilización : CALL HAC4(NAX,NBX,NCX,NH)

Parámetros utilizados :

NAX,NBX,NCX Sumandos de una operación ,en la que se busca si ha habïdo un 'HALF CARRY'

N1,N2,N3 Mitades inferiores de los bytes sumandos

NR Resultado de una suma de N1,N2,N3

NH Bandera de 'HALF CARRY'

Datos de entrada : NAX,NBX,NCX

Datos de salïda : NH

Funciones : Busca el valor de la bandera 'HALF CARRY' en operaciones de suma.

Diagrama de Flujo : VER fig. 4-4-40

Listado : Ver Apéndice A40

SUBRUTINA CR4

Definición : SUBROUTINE CR4 (MCD,NC)

Objetivo : Obtención del valor de la bandera 'CARRY'

Forma de utilización : CALL CR4 (MCD,NC)

Parámetros utilizados :

MCD Número que va a ser analizado para ver si ha habido 'CARRY'.

NC Bandera C 'CARRY'

Datos de entrada : MCD

Datos de salida : NC

Funciones : Se utiliza en operaciones del M6800 que afectan a la bandera 'CARRY' .

Diagrama de Flujo : Ver fig. 4-4-41

Listado : Ver Apéndice A41

SUBRUTINA BORR4

Definición : SUBROUTINE BORR4 (NAX,KBX,KCX,NC)

Objetivo : Obtener el valro de la bandera C 'BORROW' (llevar) en operaciones de resta.

Forma de utilización : CALL BORR4 (NAX,KBX.,KCX,NC)

Parámetros utilizados :

NAX,KBX,KCX Operandos de una instrucción de resta del M6800

NC Bandera 'CARRY' que para la resta es 'BORROW'.

Datos de entrada : NAX,KBX,KCX

Datos de salida : NC

Funciones : Se utiliza en operaciones de resta del M6800 que afectan a la bandera 'CARRY'.

Daiagrama de Flujo : Ver fig. 4-4-42

Listado : Ver Apéndice A42

SUBRUTINA NE4

Definición : SUBROUTINE NE4 (NRD,NN)

Objetivo : Obtener el valor de la bandera N (NEGATIVE).

Forma de utilización : CALL NE4 (NRD,NN)

Parámetros utilizados :

NRD Resultado de una operación del M6800 al que se analiza si produce o no un cambio en la bandera N.

NN Bandera N del M6800

Datos de entrada : NRD

Datos de salida : NN

Funciones : Busca si el resultado de una operación es o no negativo y según eso ,pone un 1 o un 0 en la bandera N.

Diagrama de Flujo : Ver fig. 4-4-43

Listado : Ver Apéndice A43

SUBRUTINA CER4

Definición : SUBROUTINE CER4 (NRD,NZ)

Objetivo : Obtener el valor de la bandera ('ZERO')

Forma de utilización : CALL CER4 (NRD,NZ)

Parámetros utilizados :

NRD Resultado de una operación del M6800,del que se averigua si produce o no cambio en la bandera Z.

1 Sí NRD=0

NZ

0 Sí NRD≠0

Datos de entrada : NRD

Datos de salida : NZ

Funciones : Pérmite ver si el resultado de una operación es cero o no.

Daiagrama de Flujo : Ver fig. 4-4-44

Listado : Ver Apéndice A44

SUBRUTINA SOBR4

Definición : SUBROUTINE SOBR4(NAX,NBX,NCX,NRD,NV)

Objetivo : Obtener el valor de la bandera V ('OVERFLOW')

Forma de utilización : CALL SOBR4 (NAX,NBX,NCX,NRD,NV)

Parámetros utilizados :

NAX,NBX,NCX,NRD Operandos en una instrucción que produzcaa un cambio en la bandera V.

NRD Resultados de una operación que afecta la bandera V.

NV Bandera de 'Overflow'.

Datos de entrada : NAX,NBX,NCX,NRD

Datos de salida : NV

Funciones : Esta subrutina permite ver si ha habido 'OVERFLOW' en las operaciones del M6800.

Diagrama de Flujo : Ver fig. 4-4-45

Listado : Ver Apéndice A45

SUBRUTINA CONB4

Definición : SUBROUTINE CONB4 (L8,IPD)

Objetivo :Transformar un número binario (8bits) a su equivalente decimal.

Forma de utilización : CALL CONB4 (L8,IPD)

Parámetros utilizados :

L8(8) 8 bits de un número en base binaria.

IPD Número decimal.

M Exponente de 2 ,para multiplicar por cada bit.

Datos de entrada : L8(8)

Datos de salida : IPD

Funciones : Se utiliza esta subrutina para las operaciones del M6800 que trabajam con bits.

Diagrama de Flujo : Ver fig. 4-4-46

Listado : Ver Apéndice A46

SUBRUTINA ARRE4

Definición : SUBROUTINE ARRE4(NCCR,KCCRD)

Objetivo : Obtener el valor decimal del bit menos significativo de un número de 8 bits.

Subprogramas llamados : CONB4

Forma de utilización : CALL ARRE4 (NCCR,KCCRD)

Parámetros utilizados :

NCCR(8) 8 bits de un número en base binaria.

KCCR(8) Arreglo auxiliar para el bit menos significativo y los demás ceros.

KCCRD Número decimal igual al primer bit.

Datos de entrada : NCCR(8)

Datos de salida : KCCRD

Funciones : Se utiliza esta subrutina en instrucciones que incluyen el valor de 'CARRY' como operando.

Daiagrama de Flujo : Ver fig. 4-4-47

Listado : Ver Apéndice A47

SUBRUTINA LOGN4

Definición: SUBROUTINE LOGN4(LL8,NND)

Objetivo: Obtener un número decimal equivalente a un arreglo lógico de bits.

Subprogramas llamados: CONB4

Forma de utilización: CALL LOGN4(LL8,NND)

Parámetros utilizados:

LL8(8) : Arreglo de valores lógicos correspondientes a los bits de un número.

AX : Valor lógico = TRUE, utilizado para probar los valores de LL8.

N8(8) : 8 bits obtenidos a partir de LL8.

NND : Número decimal obtenido.

Datos de entrada: LL8

Datos de salida: NND.

Funciones: Se utiliza en operaciones lógicas del M6800.

Diagrama de flujo: ver fig. 4-4-48 Listado: Apéndice A48.

SUBRUTINA SOB4

Definición: SUBROUTINE SOB4(NRD, NV)

Objetivo: Obtener el valor de la bandera V (overflow) en un caso especial.

Forma de utilización: CALL SOB4(NRD,NV)

Parámetros utilizados:

NRD: Resultado de una operación que se analiza, si produce o no un cambio en la bandera V.

NV
- 0 No hubo 'overflow'.
- 1 Hubo 'overflow' en la operación.

Datos de entrada: NRD.

Datos de salida: NV.

Funciones: Esta subrutina permite conocer si ha habido o no 'overflow' en la instrucción NEG.

Diagrama de flujo: Ver fig. 4-4-49.

Listado: Ver apéndice A49.

SUBRUTINA CAR4

Defnición: SUBROUTINE CAR4(NRD,NC)

Objetivo: Obtener el valor de la bandera C en un caso especial.

Forma de utilización: CALL CAR4(NRD,NC)

Parámetros utilizados:

NRD: Resultado de una instrucción, que se analiza si produce o no un cambio en C.

NC
- 0 No hubo 'CARRY'
- 1 Si hubo 'CARRY'

Datos de entrada: NRD

Datos de salida:NC

Funciones: Se utiliza esta subrutina para conocer si existe o no 'CARRY' en la instrucción NEG.

Diagrama de flujo: Ver fig. 4-4-50.

Listado: Ver Apéndice A50.

SUBRUTINA RO4

Definición : SUBROUTINE RO4 (NN8,NCCR,KDD,NCX)

Objetivo : Realizar la rotación a la derecha de los bits de un número.

Forma de utilización : CALL RO4 (NN8,NCCR,KDD,NCX)

Parámetros utilizados :

NN8(8) 8 bits de un número a ser rotados a la derecha.

NCCR(8) Arreglo que contiene los valores de las banderas de condiciones :H,I,N,Z,V,C

KDD Valor en decimal del número rotado a la derecha.

NDX Bit menos significativo del número antes de rotar.

NCX Valor de 'CARRY' antes de la rotación .

Datos de entrada : NN8,NCCR,KDD,NCX.

Datos de salida : KDD

Funciones : Simula la instrucción ROR (Rotación a la derecha)

Diagrama de Flujo : Ver fig. 4-4-51

Listado : Ver Apéndice A51

SUBRUTINA BALD4

Definición : SUBROUTINE BALD4 (NRR,NCCR,NRD)

Objetivo : Analizar el cambio en el valor de banderas de condición en instrucciones de carga del registro índice y del 'stack-pointer'

Subprogramas llamados :COND4,CER4

Forma de utilización : CALL BALD4 (NRR,NCCR,NRD)

Parámetros utilizados :

NRR Byte menos significativo de NRD.

NCCR(8) Registros de condiciones : H,I,N,Z,V,C

NRD Registro que se analiza si produce o no cambios en NCCR.

Datos de entrada : NRR,NRD,NCCR

Datos de salida : NCCR

Funciones : Se utiliza para obtener los valores de las banderas de condición en instrucciones LD.

Daiagrama de Flujo : Ver fig. 4-4-52

Listado : Ver Apéndice A52

SUBRUTINA REMOS

Definición : SUBROUTINE REMOS (MSD)

Objetivo :Decrementar un número de dos bytes.

Forma de utilización : CALL REMOS (MSD)

Parámetros utilizados :

MSD Número de dos bytes que va a ser disminuído en 1.

MOS Complemento en dos bytes del número 1.

Datos de entrada : MSD

Datos de salida : MSD

Funciones : Se utiliza en instrucciones que a registros de dos bytes se les decrementa en 1.

Diágrama de Flujo : Ver fig. 4-4-53

Listado : Ver Apéndice A53

SUBRUTINA GUARD

Definición : SUBROUTINE GUARD(NXD,MDI,MSD,MPC,NAD,NBD,MCCRD)

Objetivo : Guardar los registros en el'stack'

Subprogramas llamados : REMOS

Forma de utilización :

CALL GUARD(NXD,MPD,MSD,IEA,NAD,NBD,MCCRD)

Parámetros utilizados :

NXD Registro índice simulado.

IEA Dirección efectiva del operando de una instrucción.

MSD Dirección del 'stack' de memoria.

MPD Contador del programa simulado.

NAD Acumulador A simulado.

NBD Acumulador B simulado.

MCCRD Registro de condiciones simulado.

MAXL,MAXH Byte menos y más significativo de NXD.

MPCL,MPCH Byte menos y más significativo de MPD.

Datos de entrada : NXD,MPD,MSD,IEA,NAD,NBD,MCCRD

Funciones : Se utiliza esta subrutina en instrucciones de interrupción,y sirve para guardar los registros simulados en el 'stack de memoria'.

Diagrama de Flujo : Ver fig. 4-4-54

Listado : Ver Apéndice A54

SUBRUTINA COC4

Definición : SUBROUTINE COC4 (NAUX,NCON,M,JBAS)

Objetivo : Convertir una variable decimal a una base especificada.

Subprogramas llamados : DECE4

Forma de utilización : CALL COC4 (NAUX,NCON,M,JBAS)

Parámetros utilizados :

NAUX Registro en base decimal

NCON(16) Caracteres que en la base dada forman el número correspondiente a NAUX.

M Número de caracteres que forman un valor de dos bytes.

JBAS Base a la cual se cambia el número.

NCO(4) Caracteres del número convertido a base 16.

Datos de entrada :NAUX,M,JBAS

Datos de salida : NCON

Funciones : Se utiliza para visualizar registros en base JBAS.

Diagrama de Flujo : Ver fig. 4-4-55 Listado : Apéndice A55

SUBRUTINA NH4

Definición : SUBROUTINE NH4 (KX,KN)

Objetivo :Modificar operandos para calcular el 'HALF CARRY'

Forma de utilización : CALL NH4(KX,KN)

Parámetros utilizados :

KX Número ingresado correspondiente a 8 bits de algún registro.

KN Número que equivale a los 4 bits menos significativos del número KX.

Datos de entrada : KX

Datos de salida : KN

Funciones : Se utiliza esta subrutina en la obtención de un número que representa la mitad inferior del byte operando;siendo requerido este npumero para el calculo de la bandera H 'half Carry'.

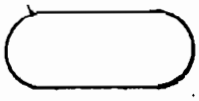
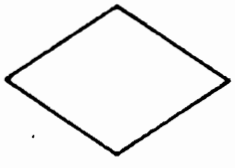
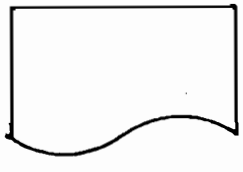
Diagrama de Flujo : Ver fig, 4-4-56

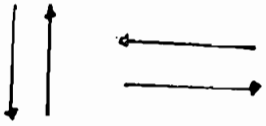
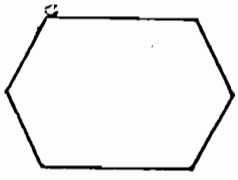
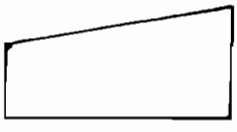
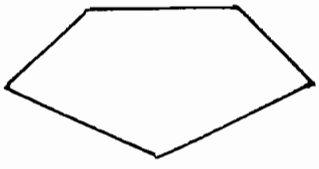
Listado : Ver Apéndice A56

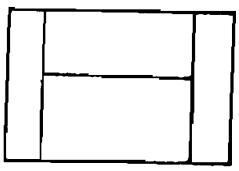
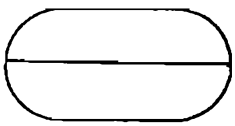
4-4 DIAGRAMAS DE FLUJO DE PROGRAMAS Y SUBRUTINAS

Los diagramas de flujo permiten representar las instrucciones que definen las operaciones y decisiones lógicas para el computador y sirven para visualizar la secuencia en los programas.

4-4-1 SIMBOLOGÍA PARA LOS DIAGRAMAS DE FLUJO

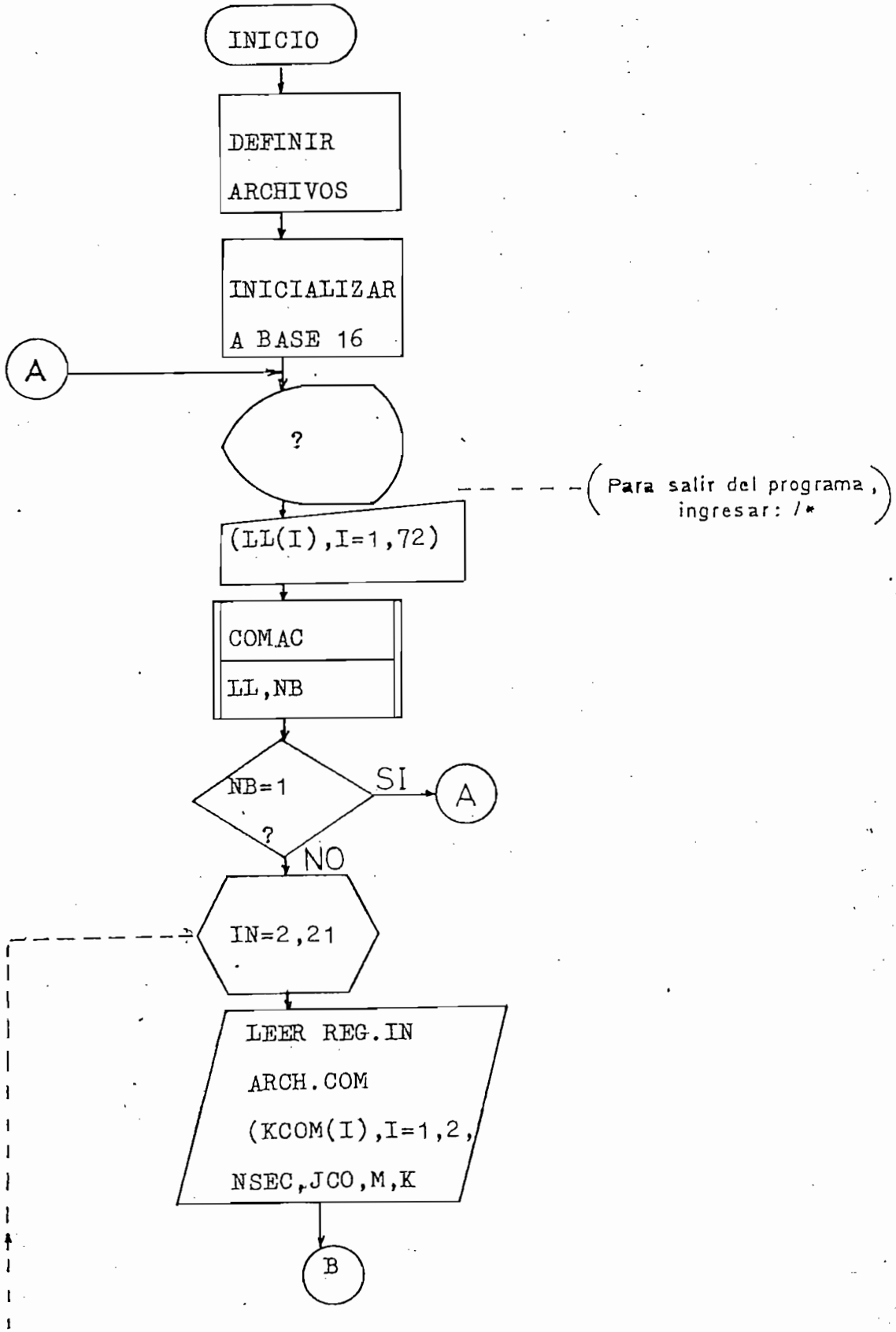
SIMBOLO	SIGNIFICADO
	Principio/fin parada de proceso
	Instrucción de bifurcación condicional.
	Instrucción de cálculo en general.
	Instrucción de impresión en papel contínuo
	Instrucción de Entrada/ Salida en general(se utilizará para disco)

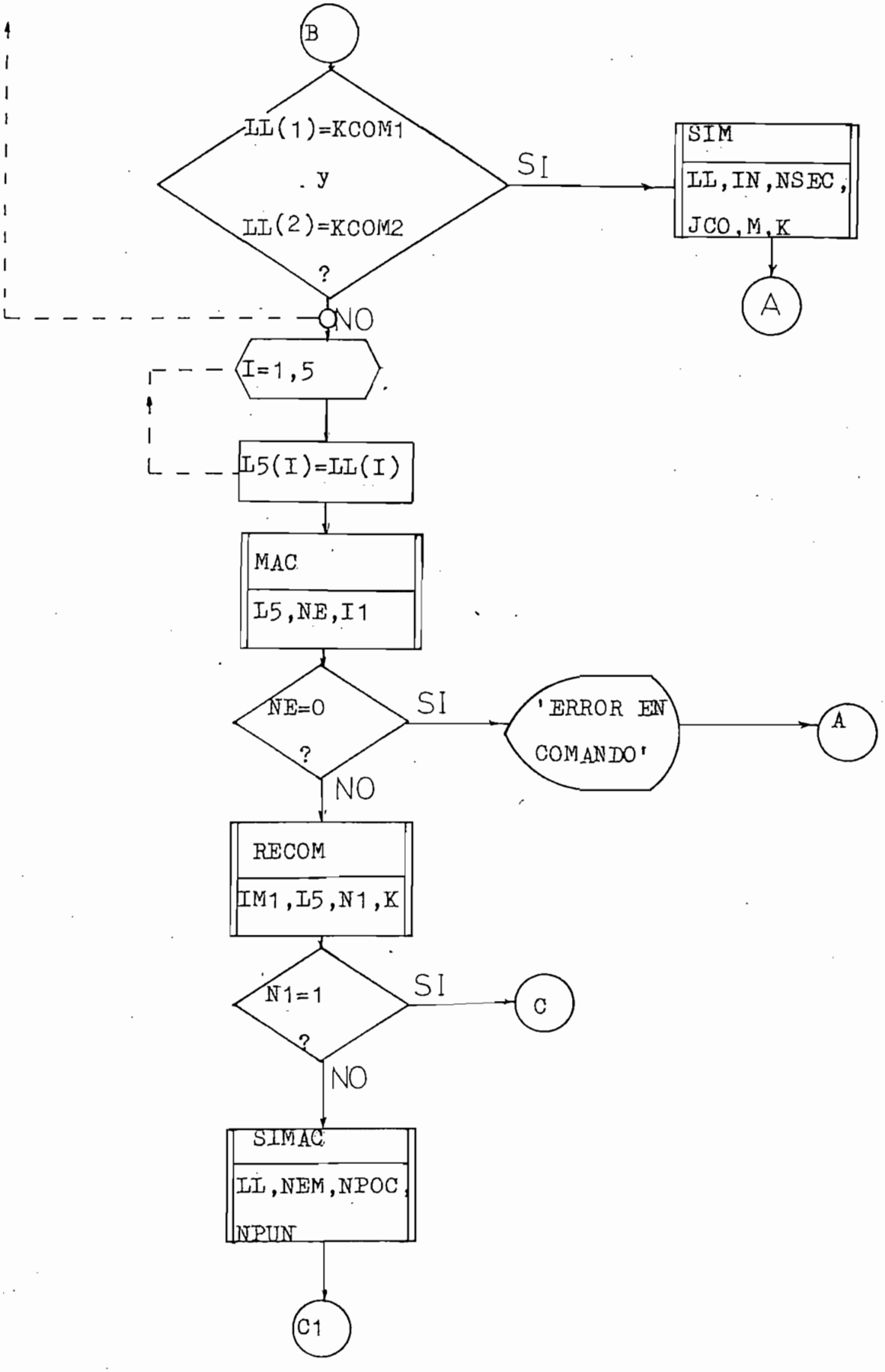
SIMBOLO	SIGNIFICADO
	Conectores entre Símbolos
	Secuencia y dirección de flujo.
	Definición de un lazo de instrucciones.
	Instrucciones de entrada de datos por teclado.
	Instrucciones de salida de Mensajes o resultados por pantalla.
	Transferencia de control de programa de acuerdo al valor de una variable.

SIMBOLO	SIGNIFICADO
	LLamada a Subrutina
	Inicio de una Subrutina

4-4-2 DIAGRAMAS DE FLUJO

-PROGRAMA S6800




B
LL(1)=KCOM1
. y
LL(2)=KCOM2
?
SI
SIM
LL,IN,NSEC,
JCO,M,K
A
NO
I=1,5
L5(I)=LL(I)
MAC
L5,NE,I1
NE=0
?
SI
'ERROR EN
COMANDO'
A
NO
RECOM
IM1,L5,N1,K
N1=1
?
SI
C
NO
SIMAC
LL,NEM,NPOC,
NPUN
C1

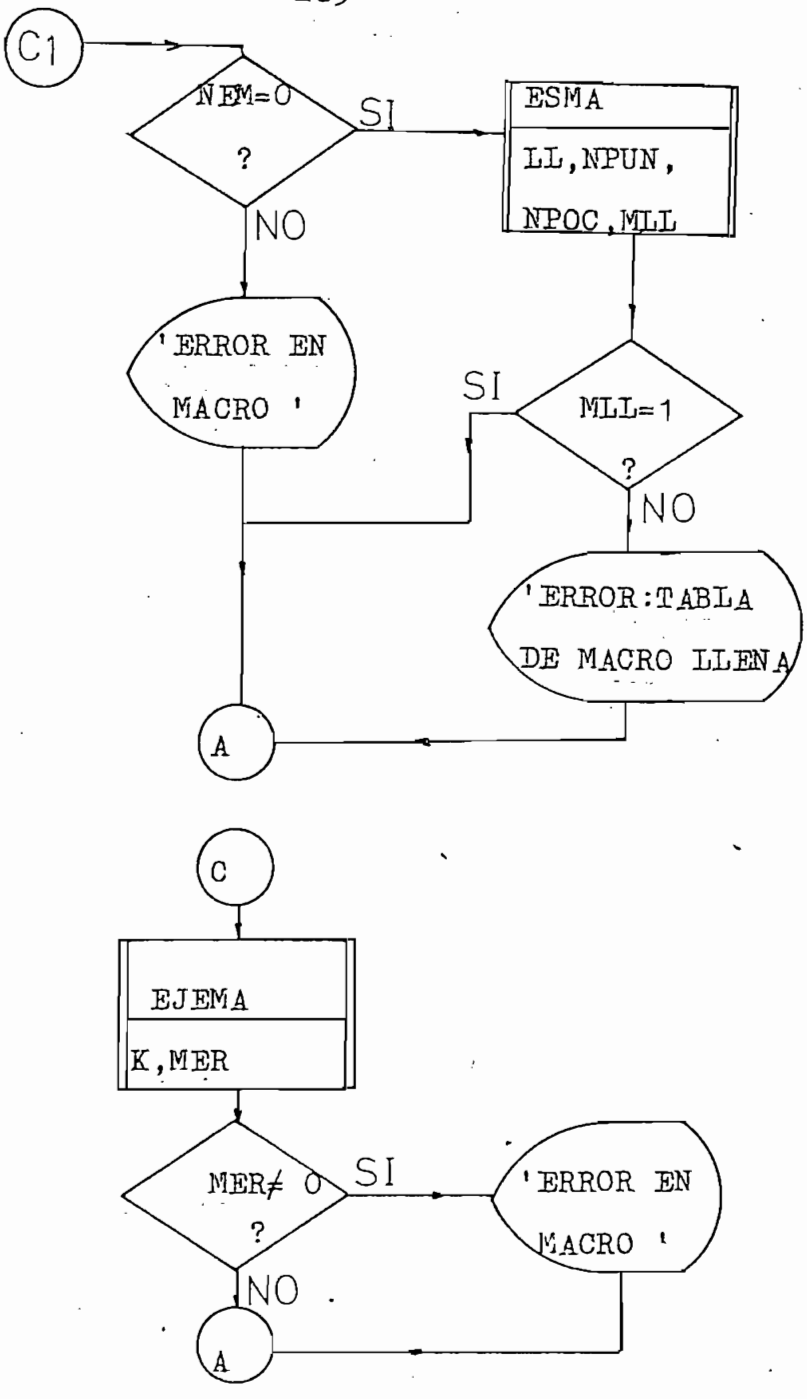

C1
NEM=0
?
SI
ESMA
LL,NPUN,
NPOC,MLL
NO
'ERROR EN
MACRO '
SI
MLL=1
?
NO
'ERROR:TABLA
DE MACRO LLENA
A
C
EJEMA
K,MER
MER≠ 0
?
SI
'ERROR EN
MACRO '
NO
A

FIG. 4-4-1

-PROGRAMA INT4

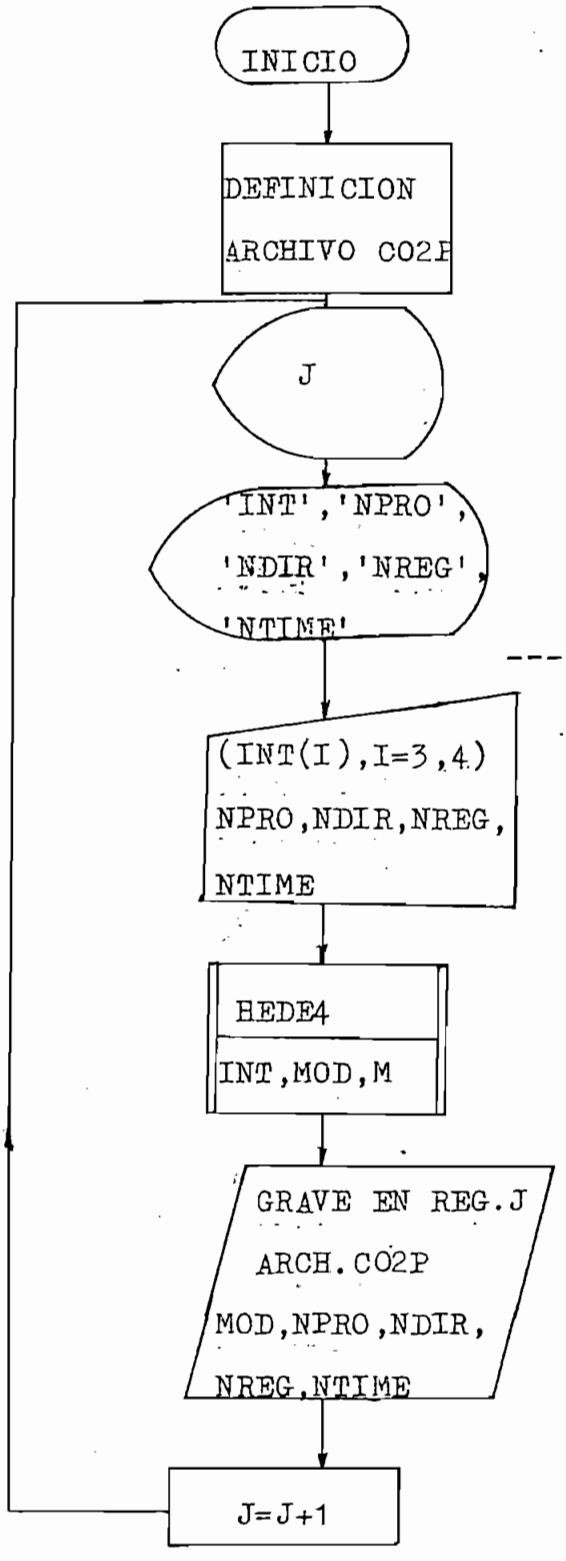


FIG. 4-4-2

-PROGRAMA LECOD

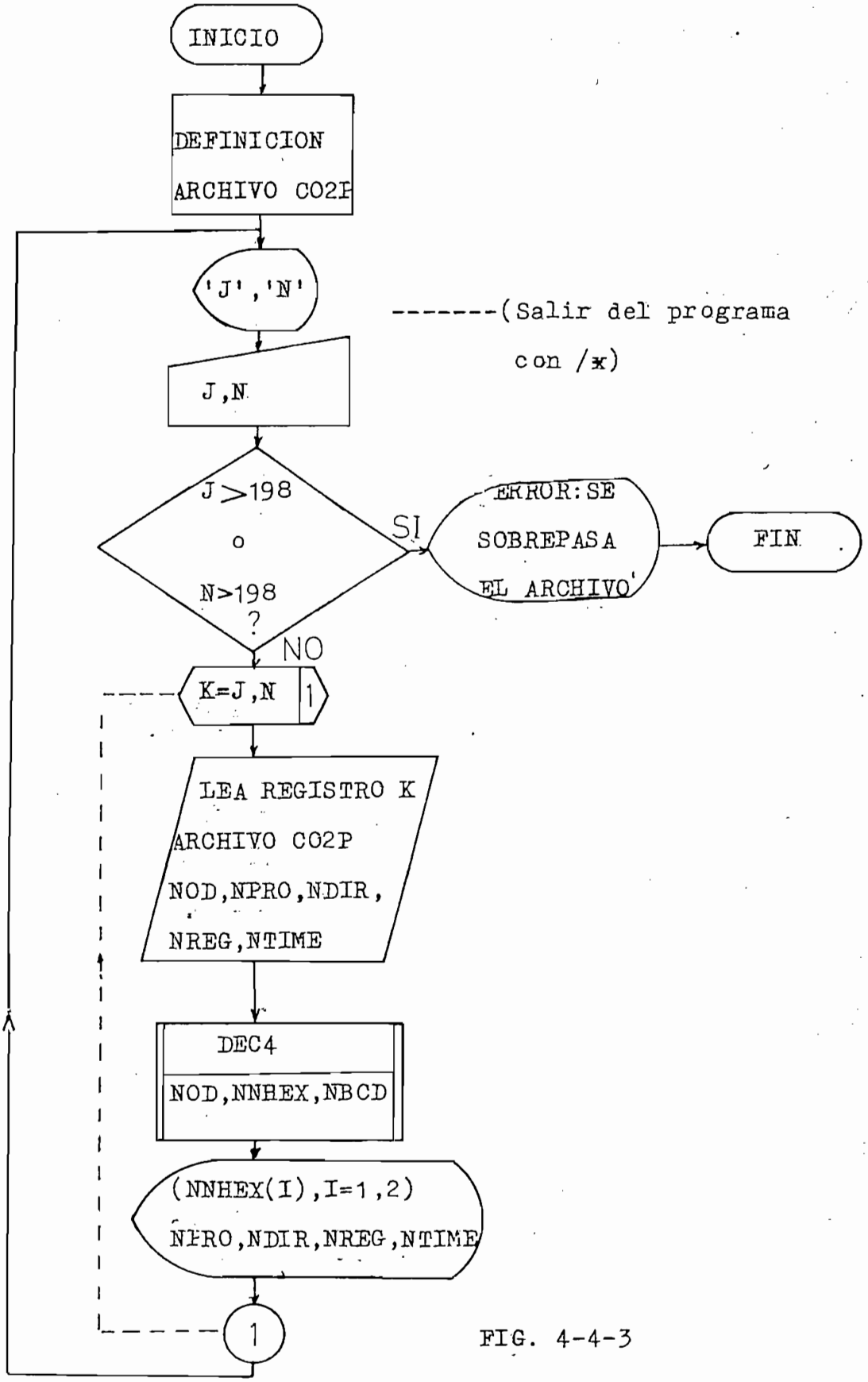
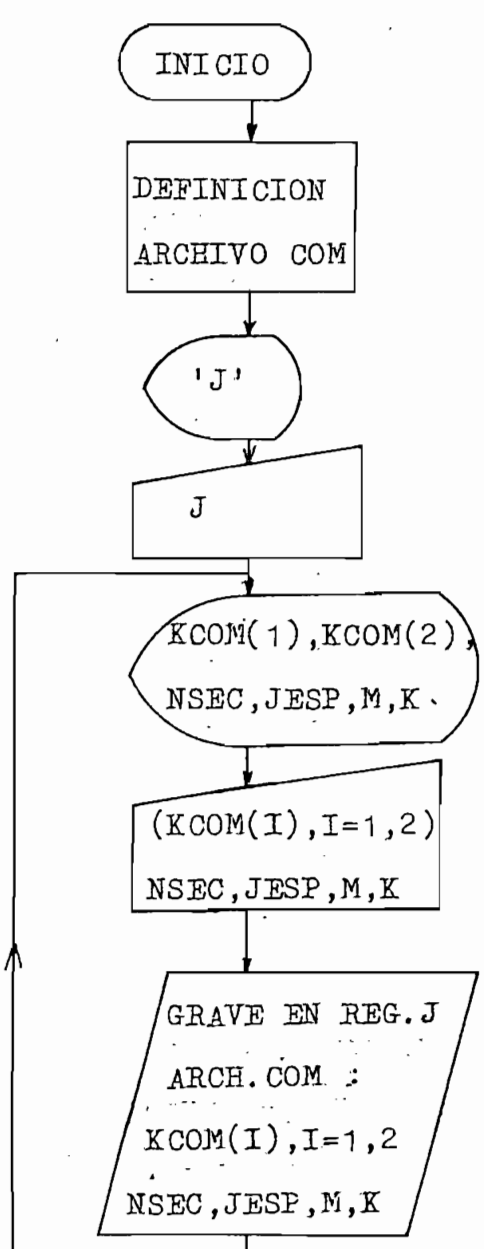


FIG. 4-4-3

-PROGRAMA COMAN



-----(Salir del programa con /*)

FIG. 4-4-4

-PROGRAMA LCOM

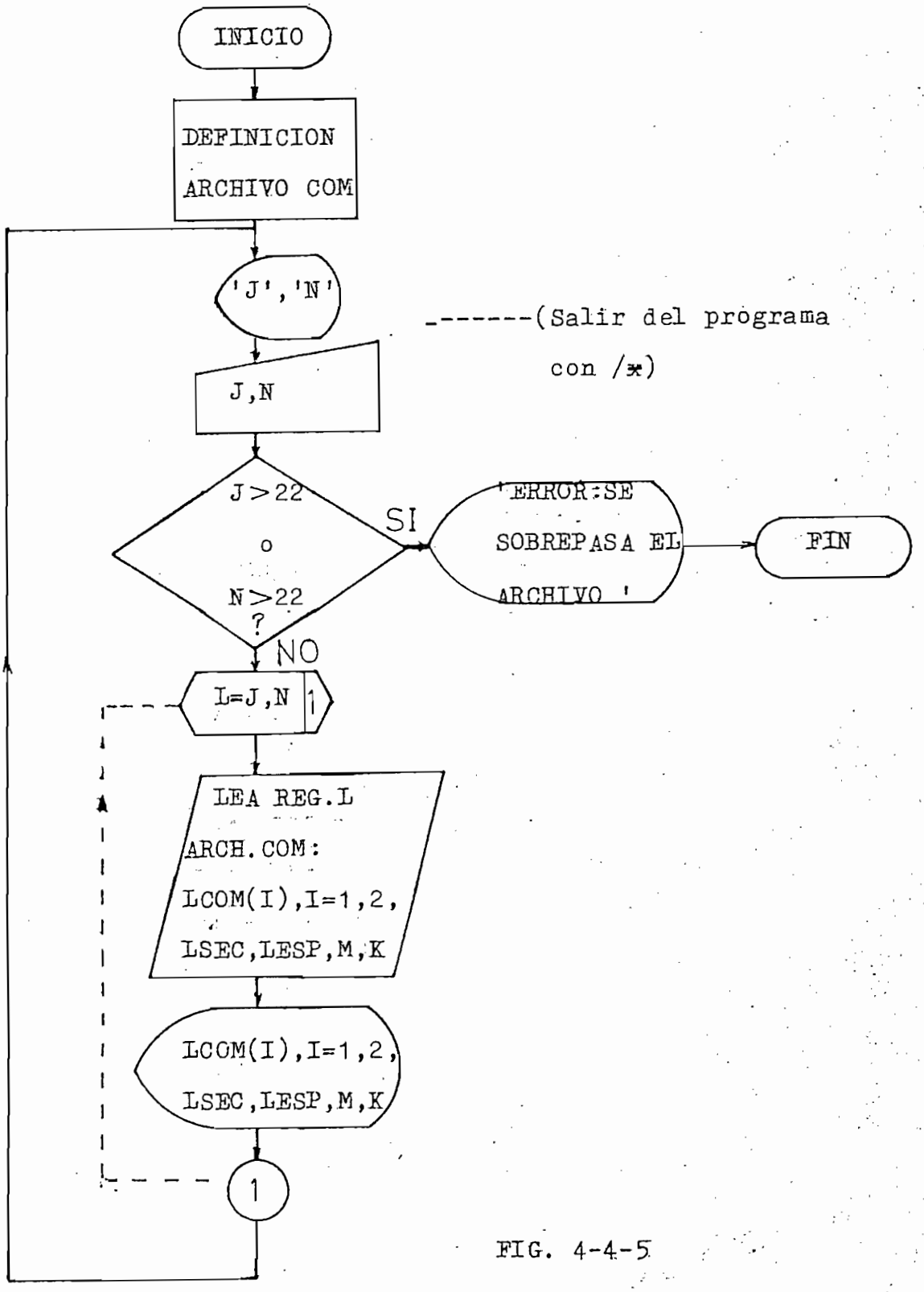


FIG. 4-4-5

-PROGRAMA CEROR

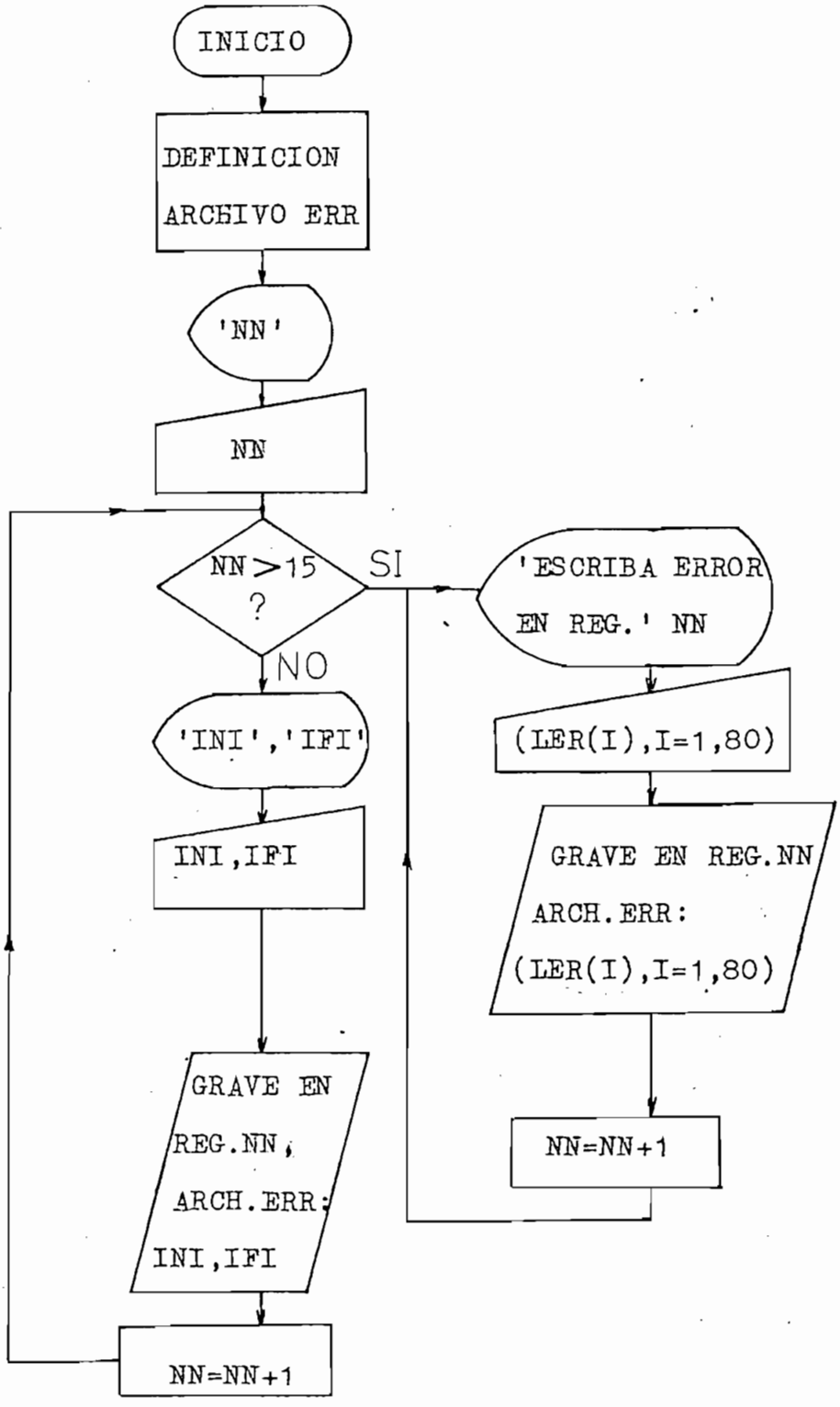


FIG. 4-4-6

-PROGRAMA LEROS

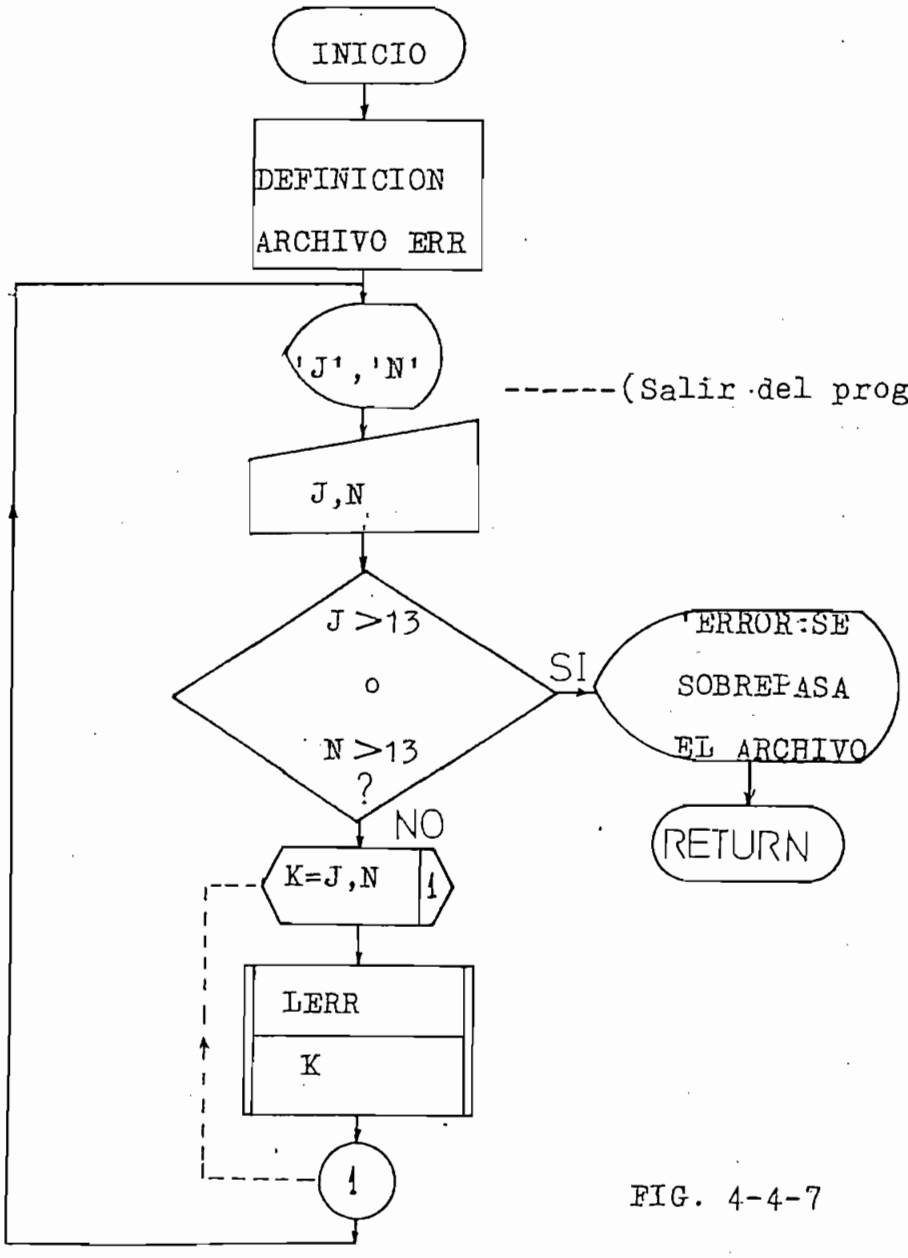
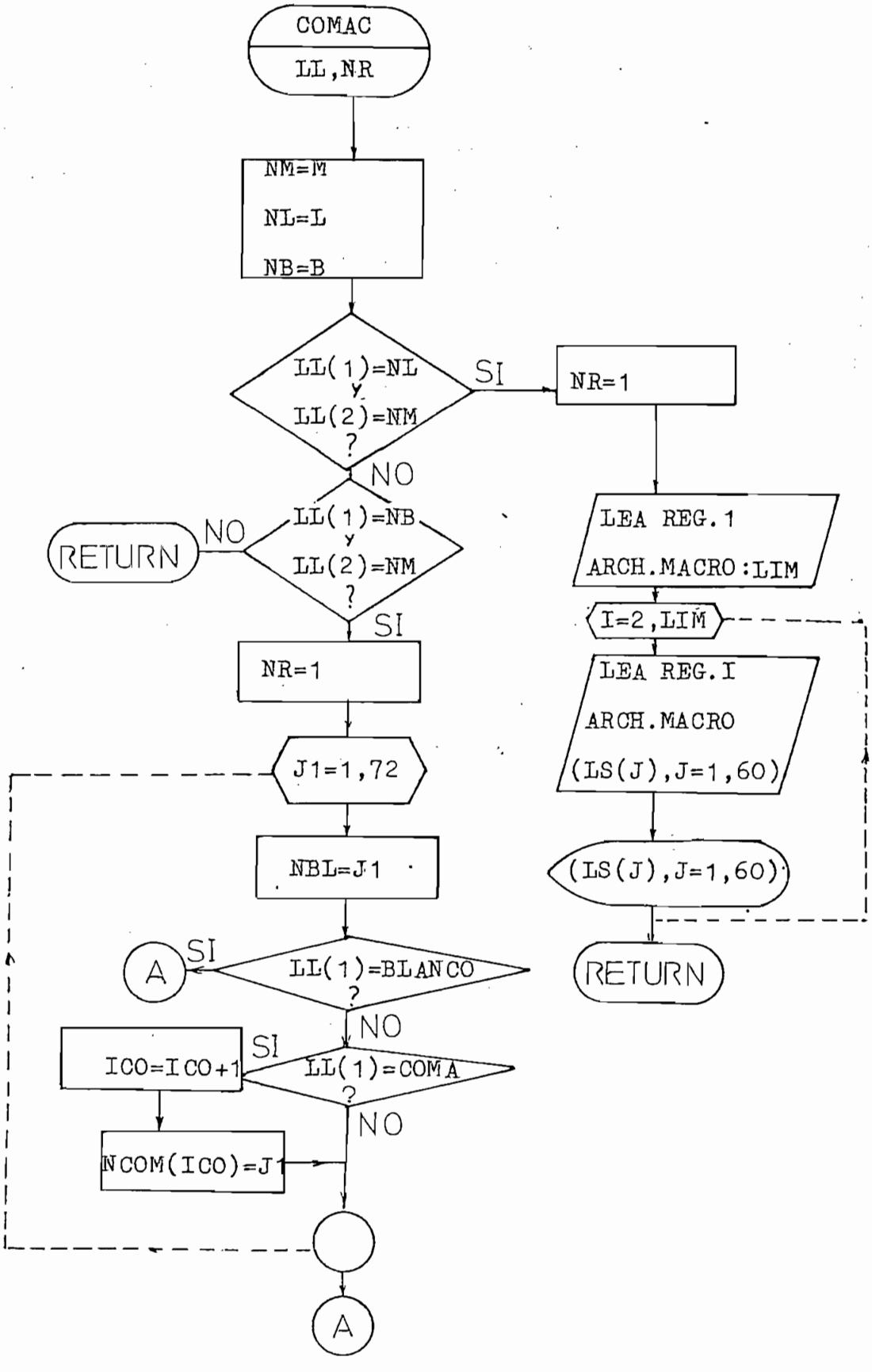


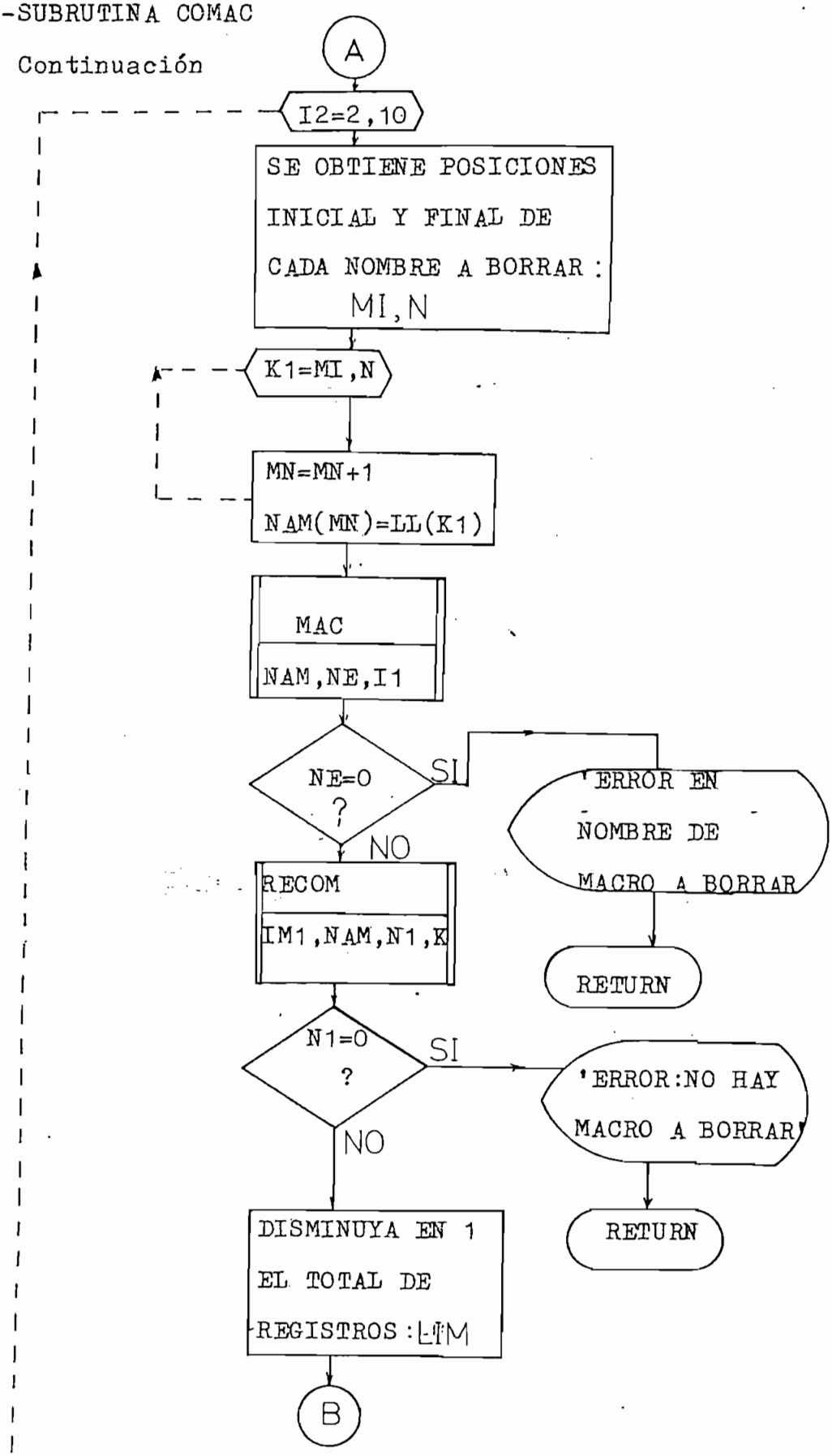
FIG. 4-4-7

-SUBRUTINA COMAC



-SUBRUTINA COMAC

Continuación



SUBRUTINA COMAC

Continuación

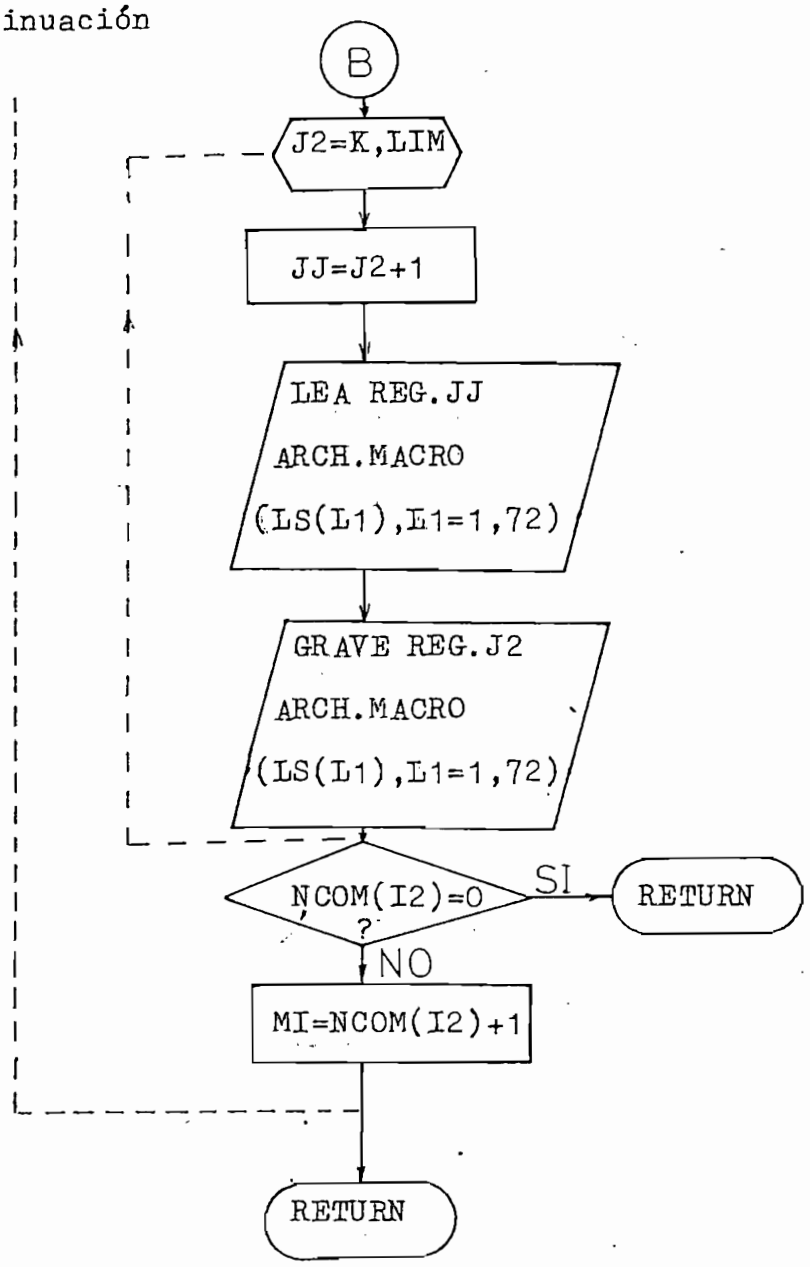
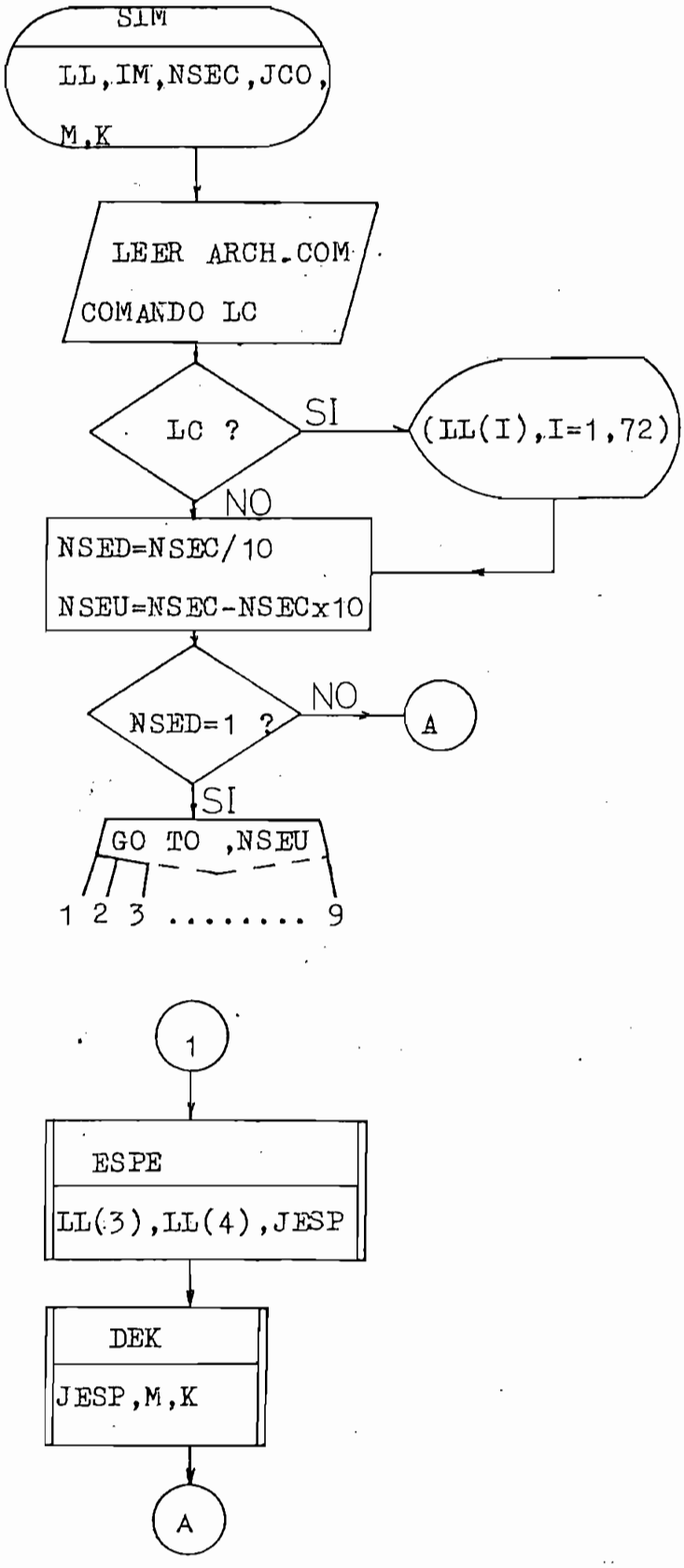


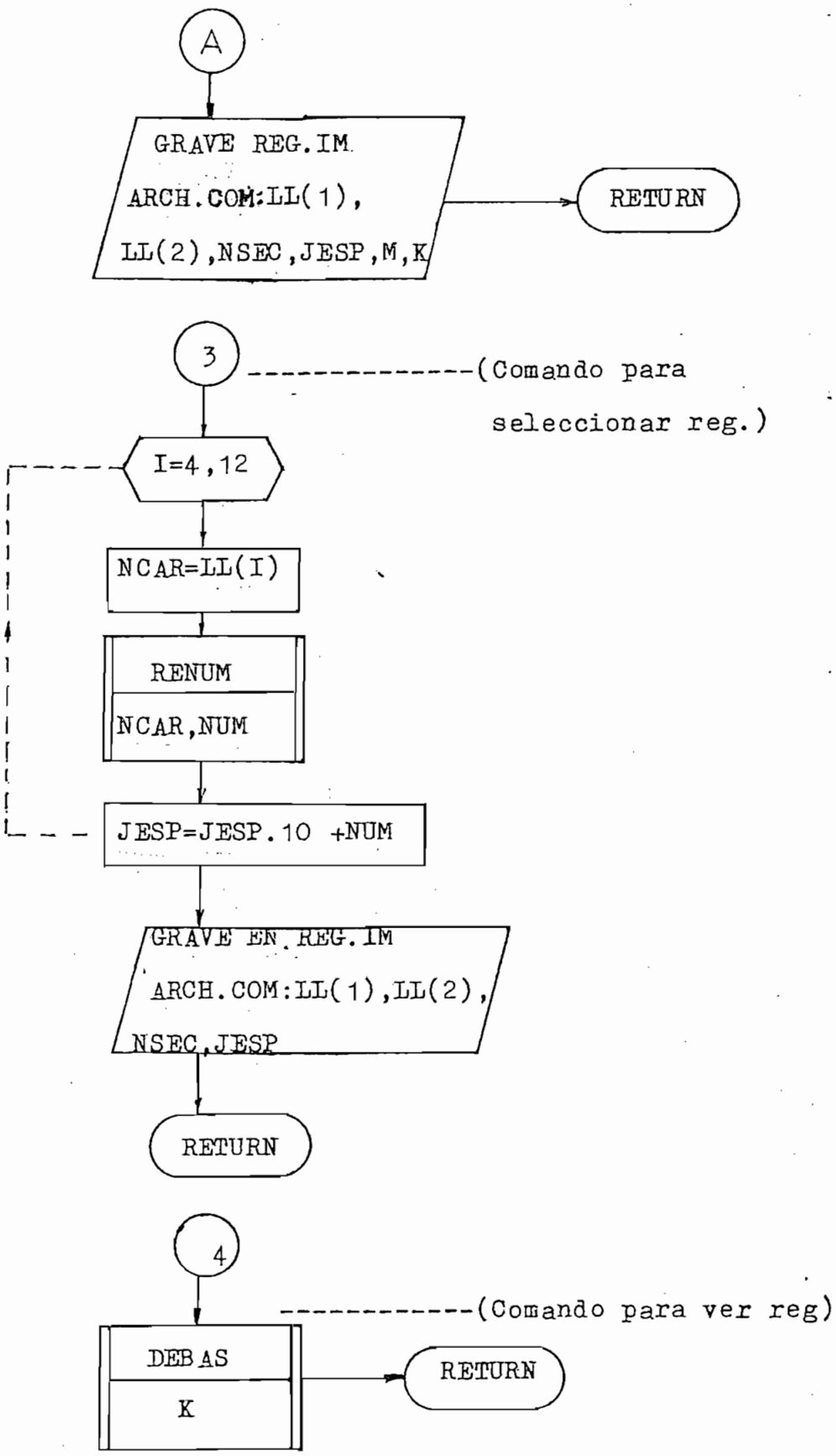
FIG. 4-4-8

-SUBRUTINA SIM



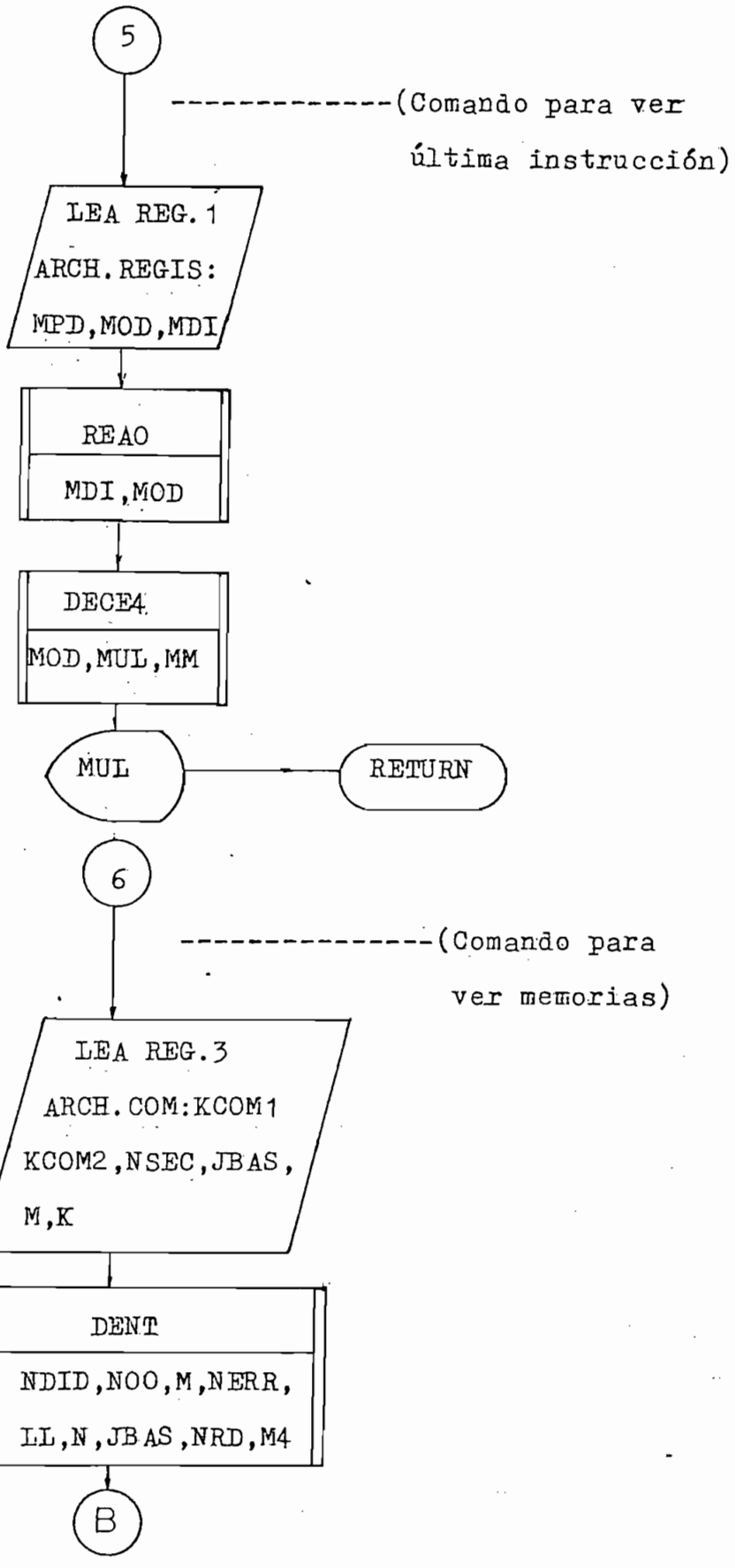
SUBRUTINA SIM

Continuación



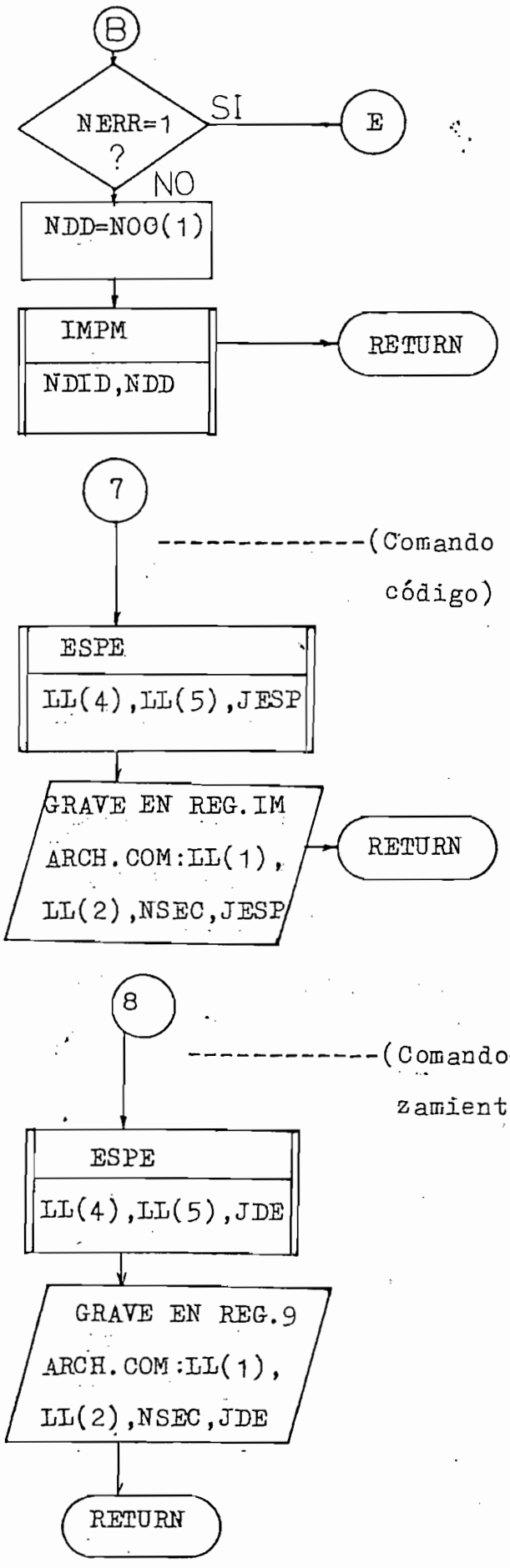
SUBRUTINA SIM

Continuación



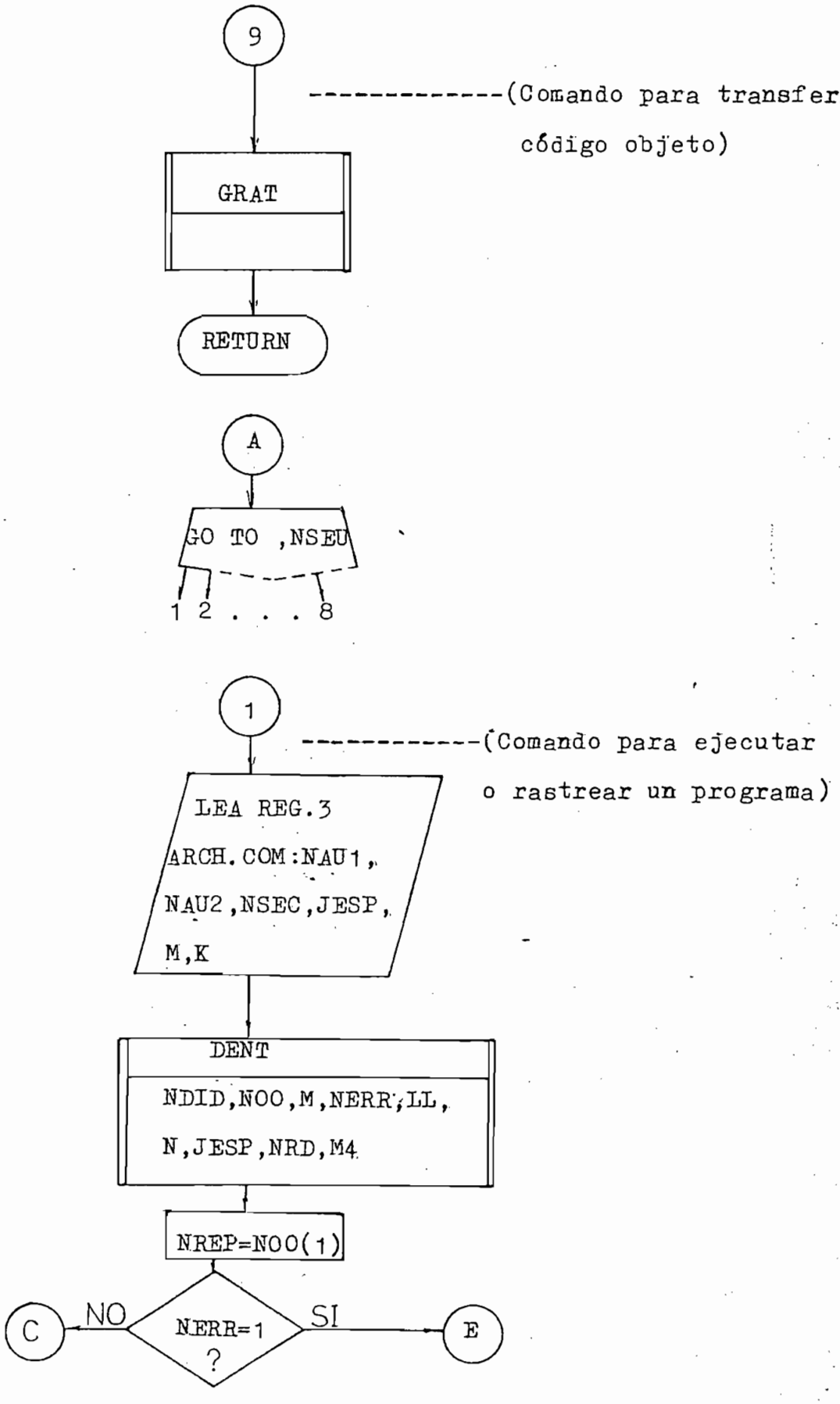
SUBRUTINA SIM

Continuación



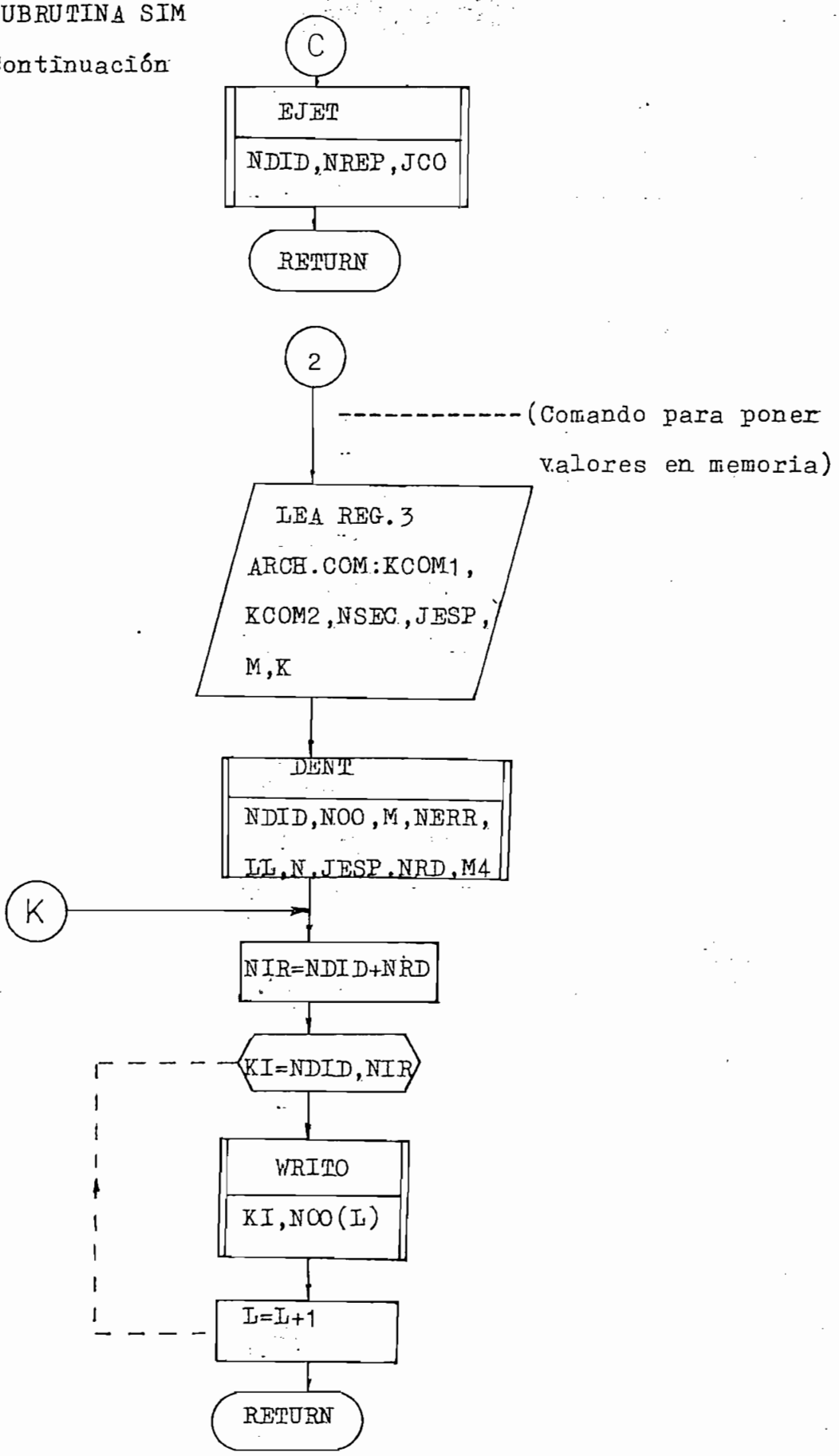
SUBRUTINA SIM

Continuación



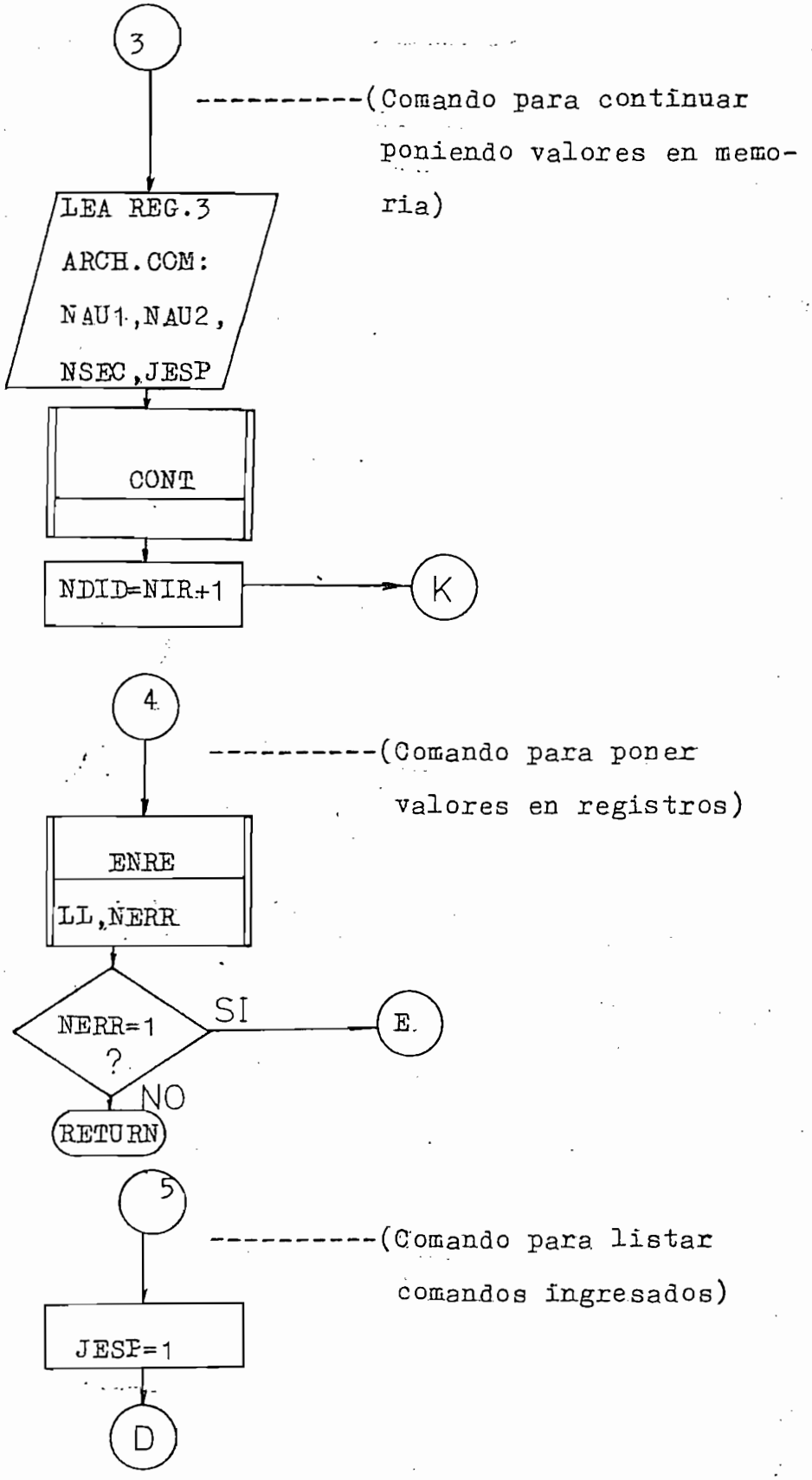
SUBRUTINA SIM

Continuación



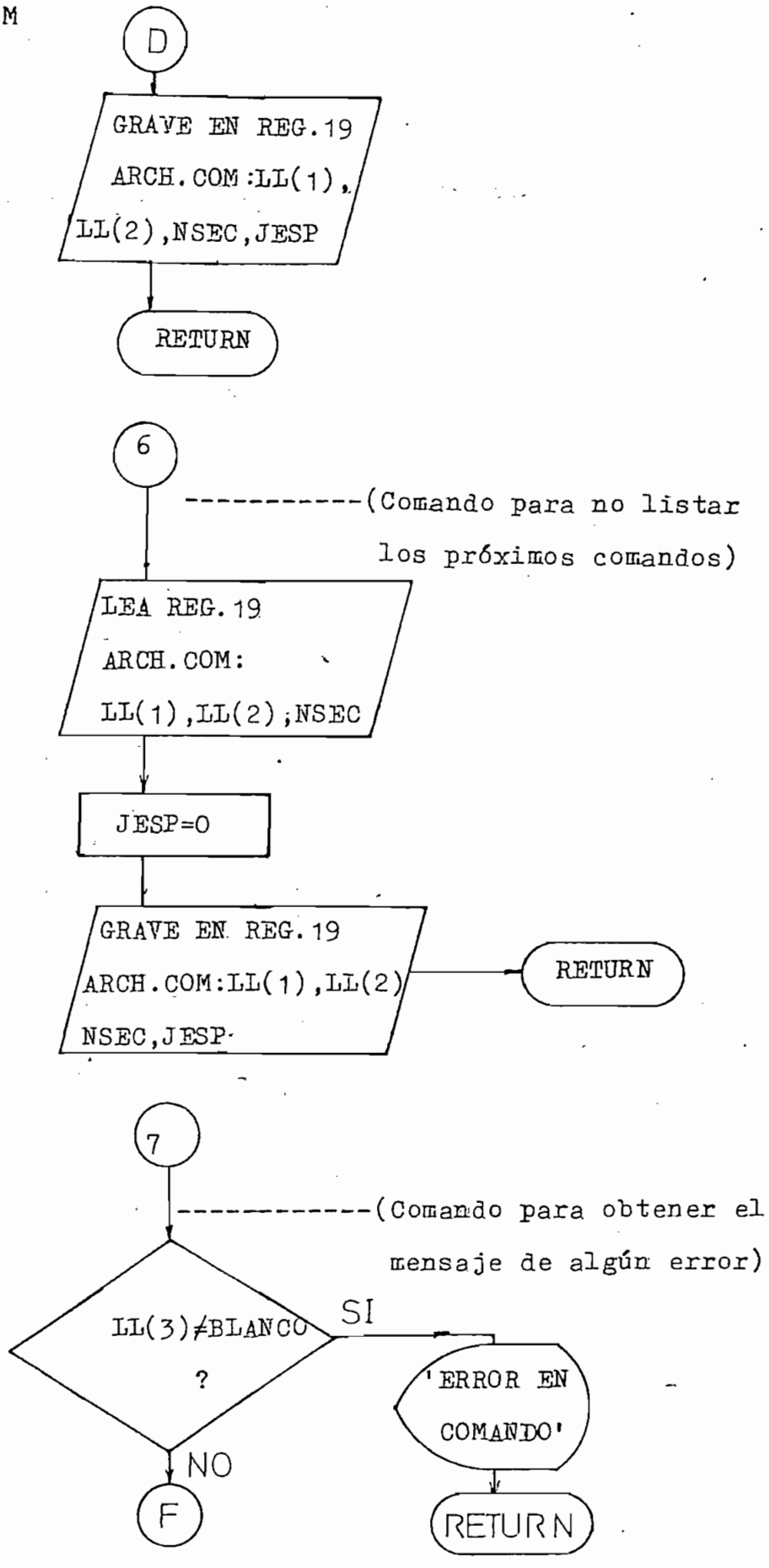
SUBRUTINA SIM

Continuación



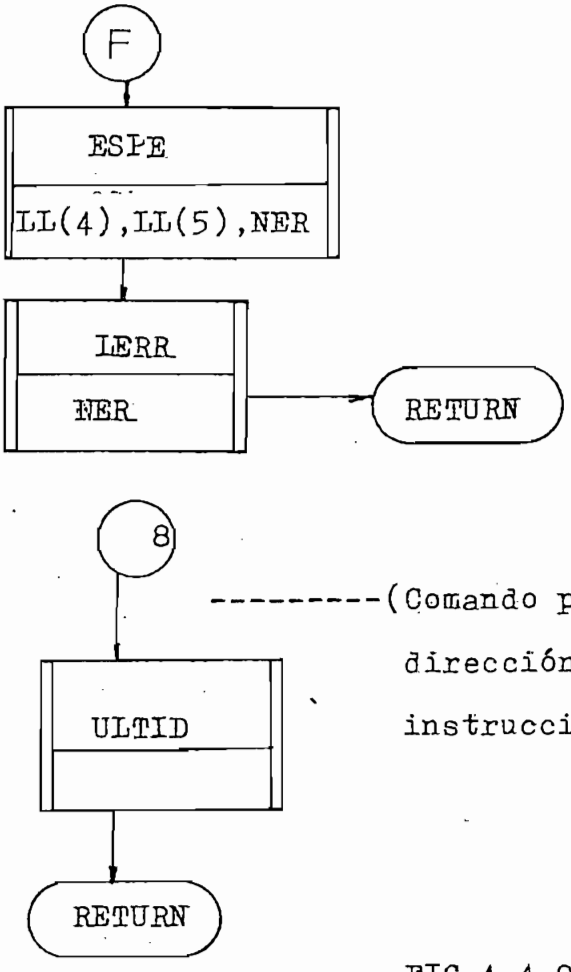
SUBRUTINA SIM

Continuación



SUBRUTINA SIM

Continuación



---------(Comando para obtener la dirección de la última instrucción)

FIG.4-4-9

SUBRUTINA MAC

Para la realización de esta subrutina se aplicará los conceptos de máquinas de estado finitas y matrices de transición expuestos en los puntos 2-2 y 2-3;cabe señalar que el objetivo de esta subrutina es el reconocimiento de macros.

Primeramente debe plantearse la gramática o conjunto de reglas que cumple un nombre o identificador de un macrocomando;estas reglas deben satisfacer las condiciones siguientes: un nombre debe iniciarse con una letra,los siguientes caracteres pueden ser letras o números.Por lo tanto si se llama al nombre de un macrocomando como identificador,las reglas quedaríam así:

<identificador> ::= letra
<identificador> ::=<identificador>letra
<identificador> ::=<identificador>dígito

Por ejemplo la 2da.regla muestra que un identificador puede contener una parte del nombre hasta este momento formada,seguida por una letra.

Ejemplos de nombres que cumplen estas reglas pueden ser: ALFA,B25E,IDEN,NAM,etc.

Se ha supuesto en los ejemplos anteriores de nombres la limitación de cuatro caracteres como máximo.

Para obtener el autómata de estados finitos que repre-

senta a esta gramática ,se diseña el diagrama de estados en base a las reglas anteriores y mediante el método expuesto en la sección 2-2-2.

Si,para formar el diagrama de estados se llama:S al estado inicial,I al estado que contiene al identificador , Z al estado final y F un estado de fallo, el diagrama quedaría así:

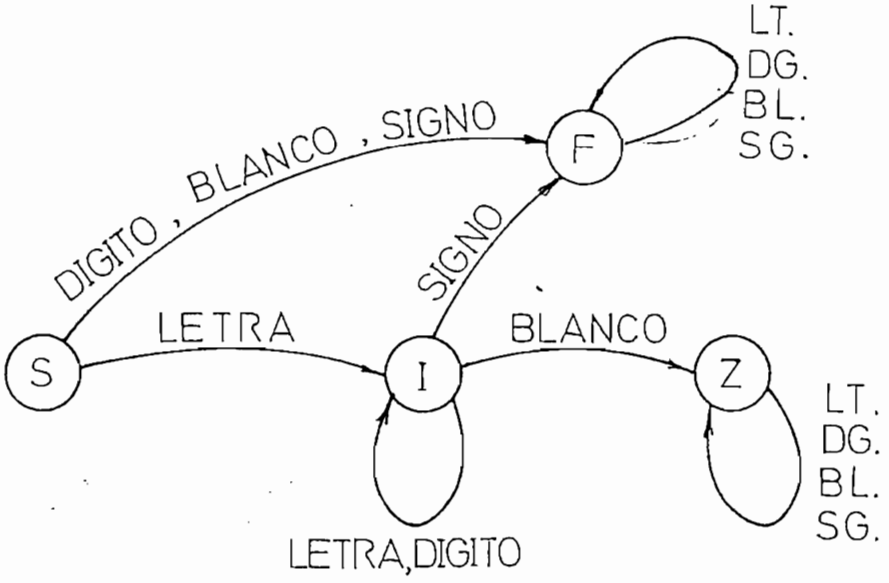


FIG. 4-4-10A

El autómata de estados finitos se deduce a partir del diagrama anterior y en base al punto 2-2,donde se decía que el autómata está formado por:

K :Un alfabeto de estados,en este caso son:S,I,F,Z

VT:Un alfabeto de entrada :Letras,Dígitos,Blancos,Signos

M :Función que pasa de un K y VT a otro K

S :Estado Inicial

Z :Estado Final

De esta manera el autómata de estados finitos es:

$$AF\left(\left\{S,I,F,Z\right\},\left\{\text{Letra},\text{Dígito},\text{Blanco},\text{Signo}\right\},M,S,\left\{Z\right\}\right)$$

M(S,Blanco)	=F	M(Z,Letra)	=Z
M(S,Letra)	=I	M(Z,Dígito)	=Z
M(S,Dígito)	=F	M(Z,Blanco)	=Z
M(S,Signo)	=F	M(Z,Signo)	=Z
M(I,Letra)	=I	M(F,Letra)	=F
M(I,Dígito)	=I	M(F,Dígito)	=F
M(I,Blanco)	=Z	M(F,Blanco)	=F
M(I,Signo)	=F	M(F,Signo)	=F

Como se había dicho la función M realiza una transición de un estado al otro, con el ingreso de algún caracter; así por ejemplo en M(S,Blanco)=F, quiere decir, que al encontrarse en el estado inicial S hay el ingreso de un caracter blanco, se pasa a un estado fallido F; esto porque un nombre debe empezar con una letra y no con un espacio en blanco.

Para formar la matriz de transición, se toma como guía lo expuesto en el punto 2-3 y se lo aplica de la siguiente manera:

Si se toma solo las iniciales de los caracteres de entrada: L para letra, D para dígito, B para un espacio en blanco, S para signo; y se coloca encabezando cada columna ; los estados: S,I,F,Z en cada fila; y se aumenta una última columna etiquetada por un signo $\dashv$, que indica los estados aceptados con un 1 y no aceptados con un 0; la matriz ya formada aparecería así:

	L	D	B	S	⊣	
S	I	F	F	F	0	
I	I	I	Z	F	0	
F	F	F	F	F	0	
Z	Z	Z	Z	Z	1	(Salida Válida)

FIG. 4-4-10 b

Con todo este desarrollo se pasa a ver como quedaría el diagrama de flujo ;para esto inicializando una matriz de 4x5 y tomando a S como la fila 1 de la matriz hasta la fila 4 de Z ; y como columnas 1 a 5 de la matriz desde L hasta ⊣ ; se dará valores a la matriz en base al gráfico anterior.

El diagrama de flujo está en la fig. 4-4-10.c en la siguiente hoja.

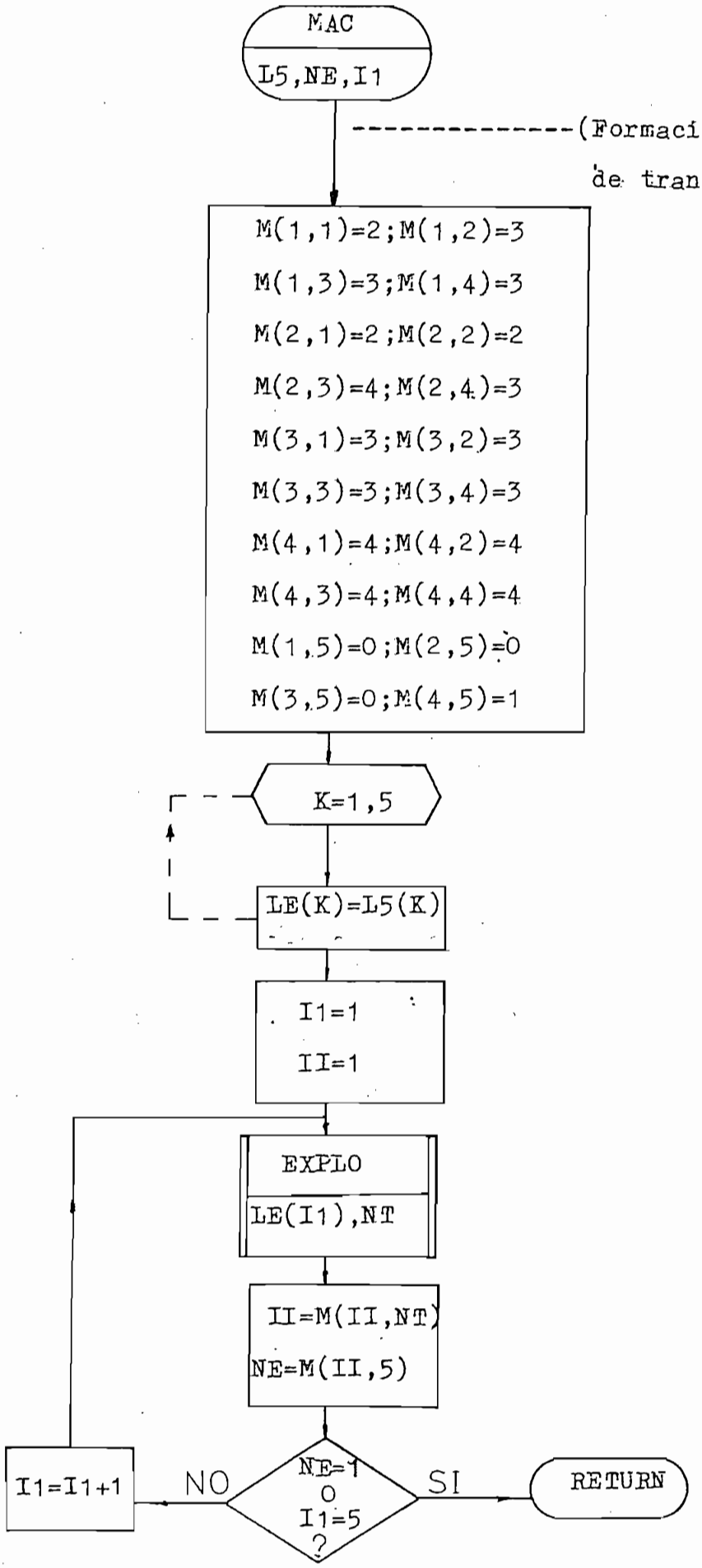

MAC
L5,NE,I1
----------(Formación de la matriz de transición)
M(1,1)=2;M(1,2)=3
M(1,3)=3;M(1,4)=3
M(2,1)=2;M(2,2)=2
M(2,3)=4;M(2,4)=3
M(3,1)=3;M(3,2)=3
M(3,3)=3;M(3,4)=3
M(4,1)=4;M(4,2)=4
M(4,3)=4;M(4,4)=4
M(1,5)=0;M(2,5)=0
M(3,5)=0;M(4,5)=1
K=1,5
LE(K)=L5(K)
I1=1
II=1
EXPLO
LE(I1),NT
II=M(II,NT)
NE=M(II,5)
NE=1 o I1=5 ?
NO
SI
I1=I1+1
RETURN

FIG.4-4-10.c

A manera de comentario de la subrutina MAC,que es realizada con la utilización de máquinas de estado finitas,se debe afirmar que ha habido una considerable ventaja en el manejo de memoria,lo mismo que una cierta facilidad para hacer cambios en el funcionamiento del programa,pues cambiando únicamente los valores de la matriz se le puede dar otras características, como por ejemplo la de aceptar que haya signos en los nombres de un macrocomando,etc.

Pero hay un inconveniente al programar de esta forma, y es : el aumento de memoria utilizada; es por esto que se ha preferido no realizar todos los programas utilizando este método ;sin embargo como ejemplo se ha realizado la subrutina MAC.

SUBRUTINA RECOM

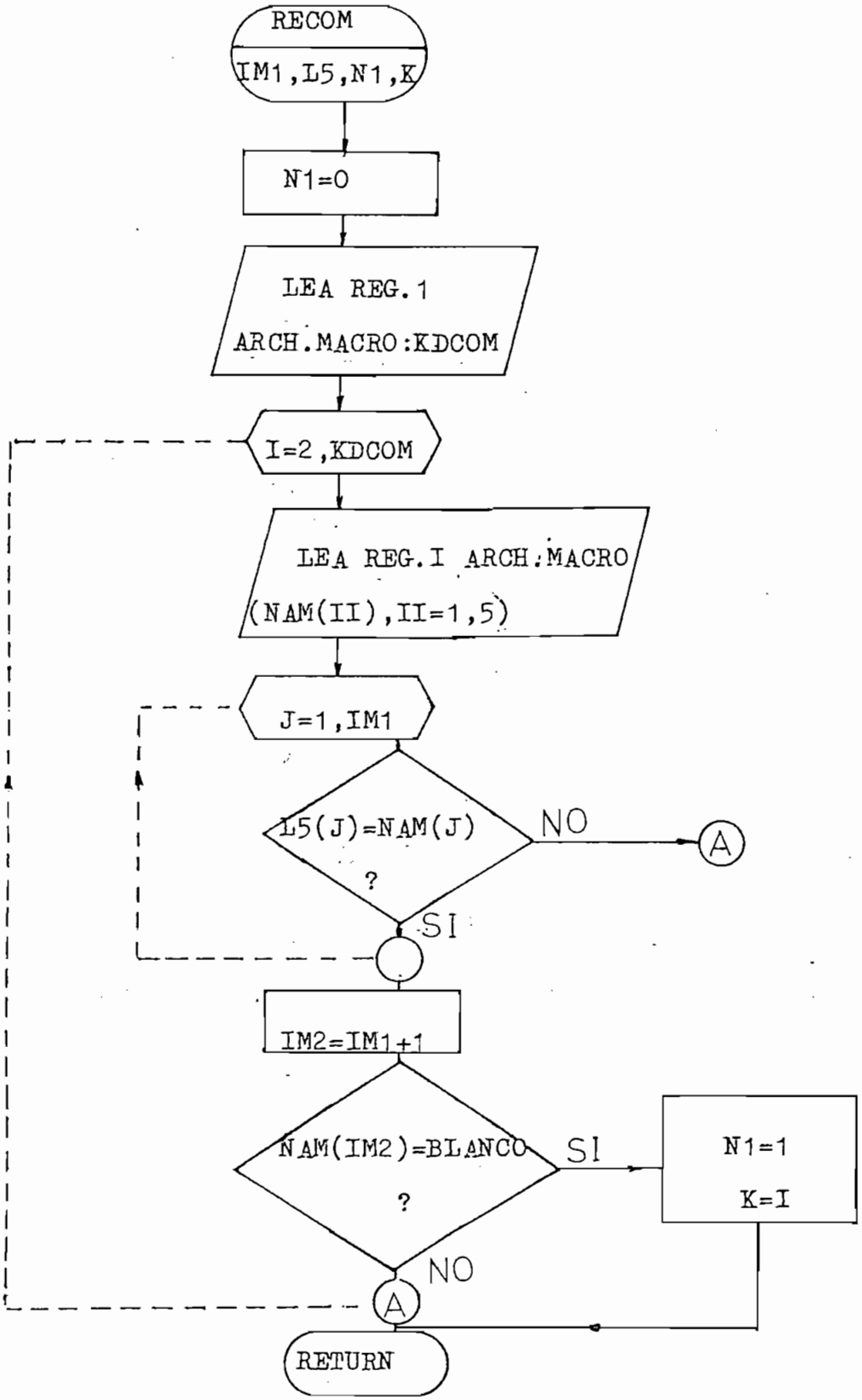


FIG.4-4-11

-SUBRUTINA SIMAC

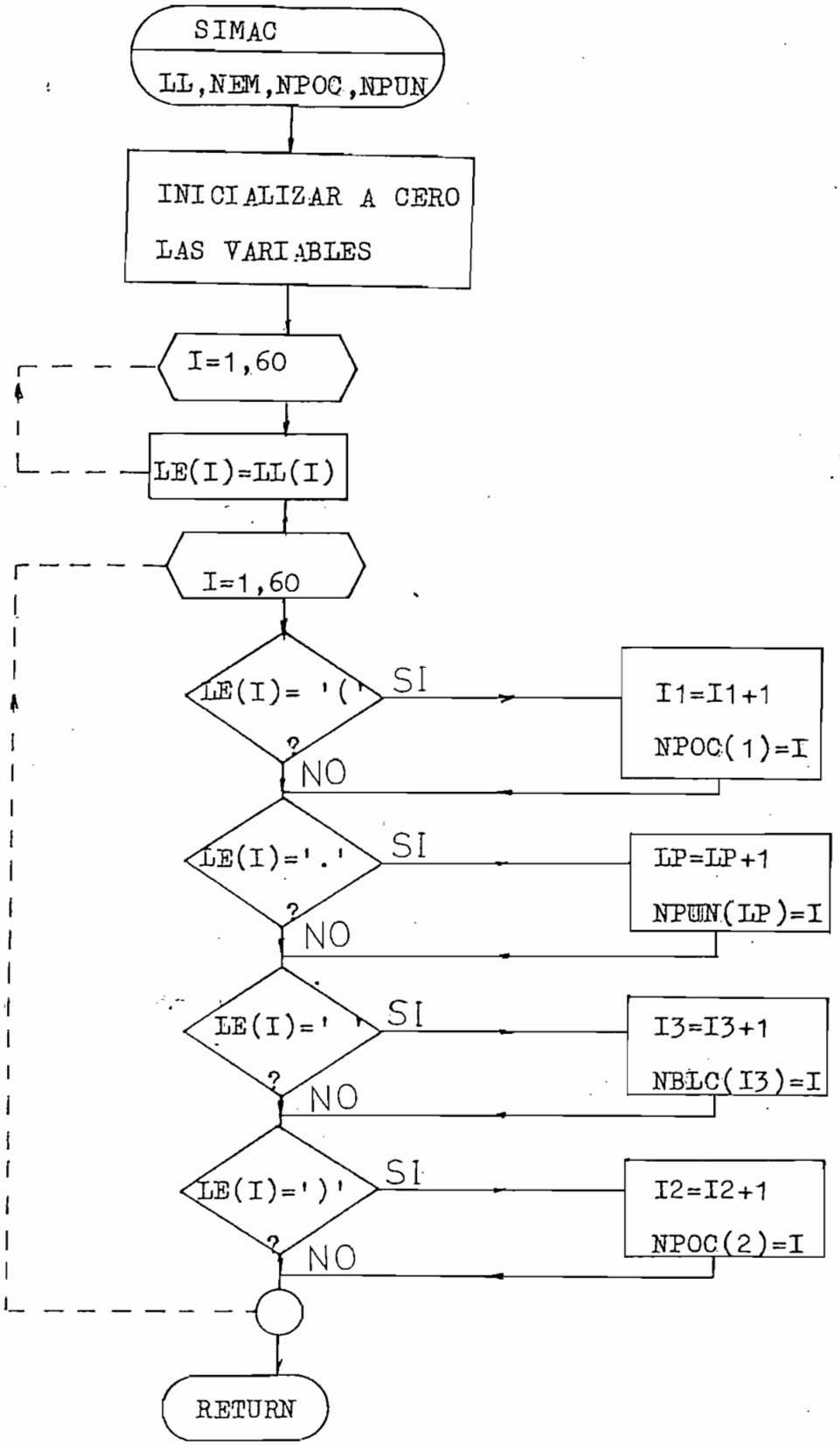
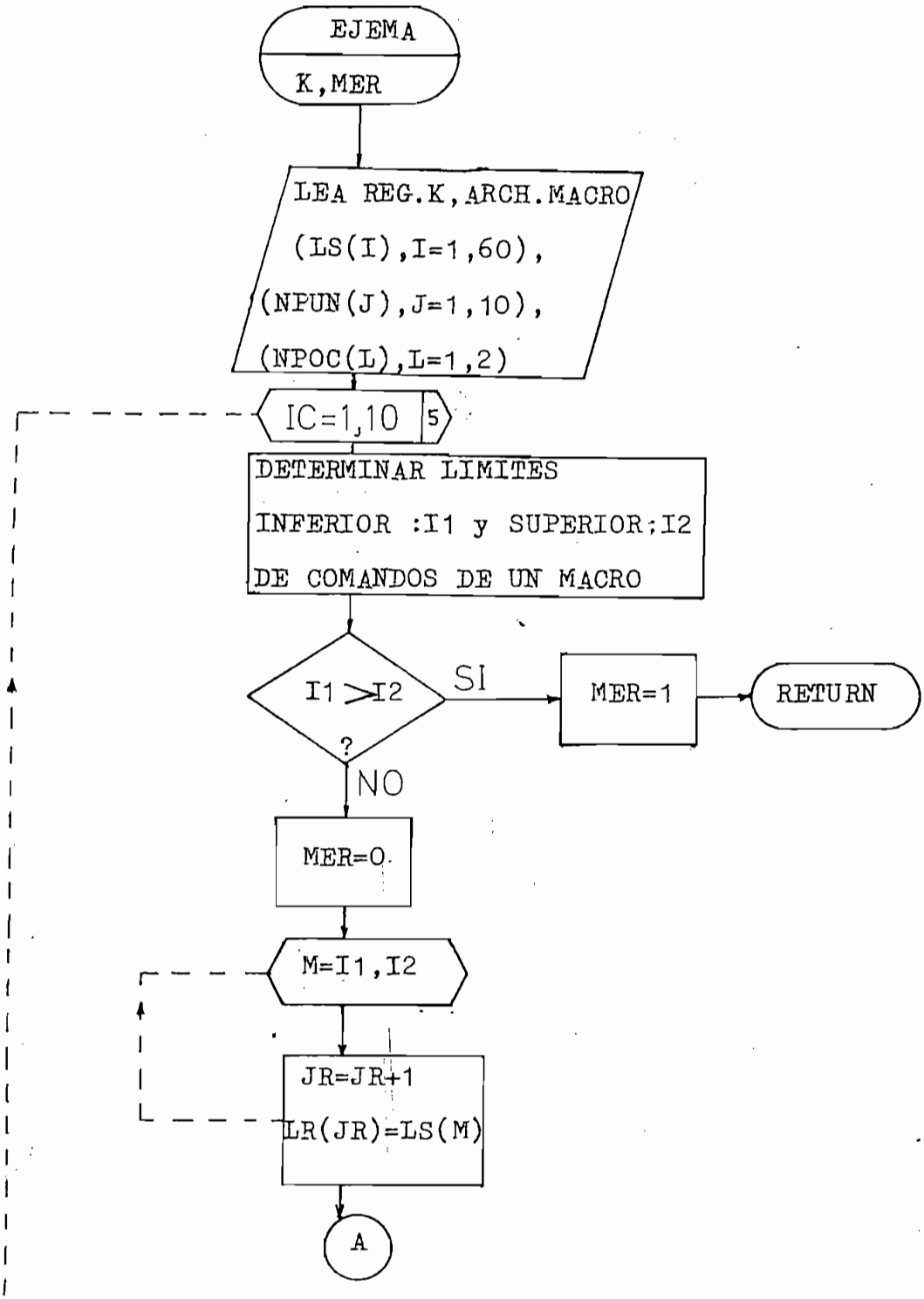


FIG. 4-4-12

-SUBRUTINA EJEMA



SUBRUTINA EJEMA

Continuación

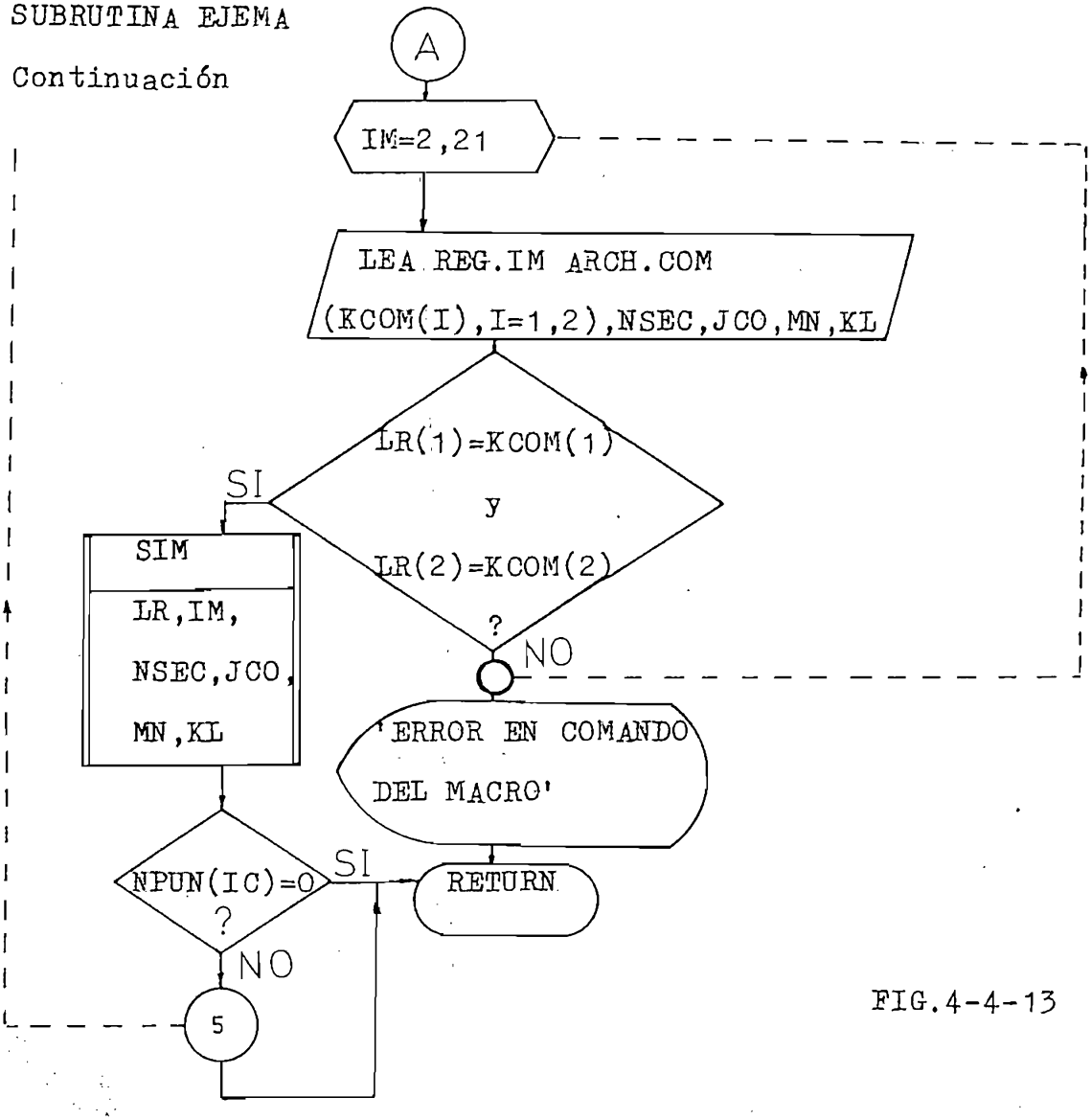


FIG.4-4-13

-SUBRUTINA ESMA

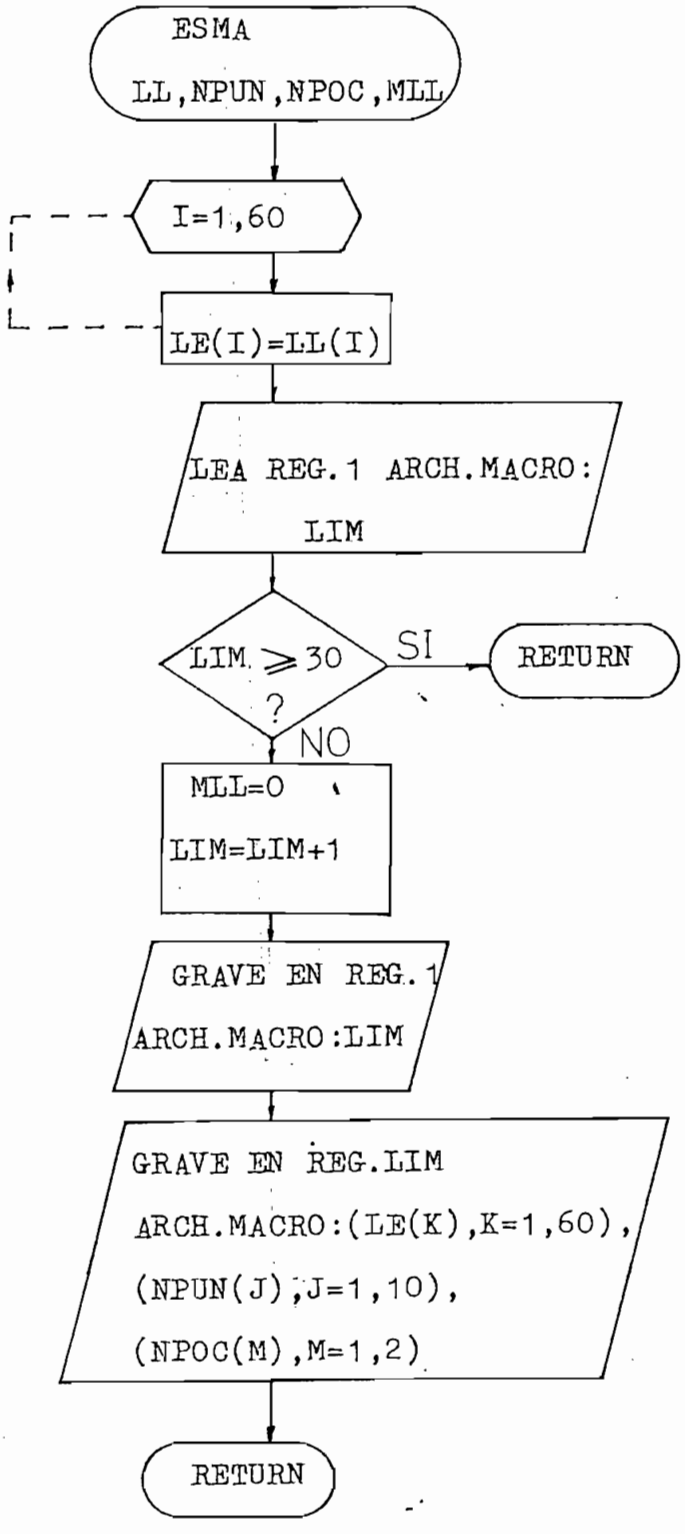


FIG. 4-4-14

SUBRUTINA HEDE4

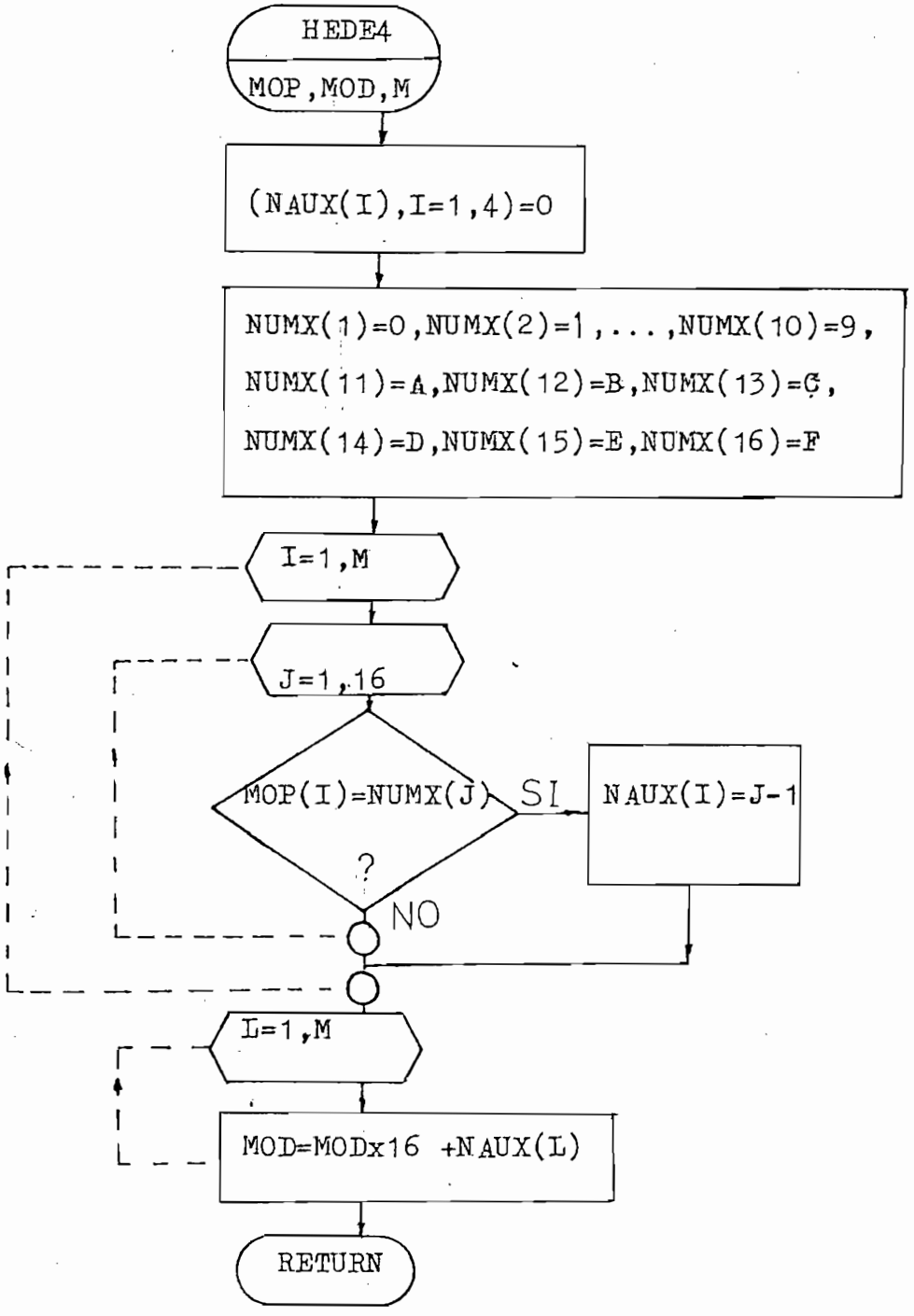


FIG. 4-4-15

SUBRUTINA DEC4

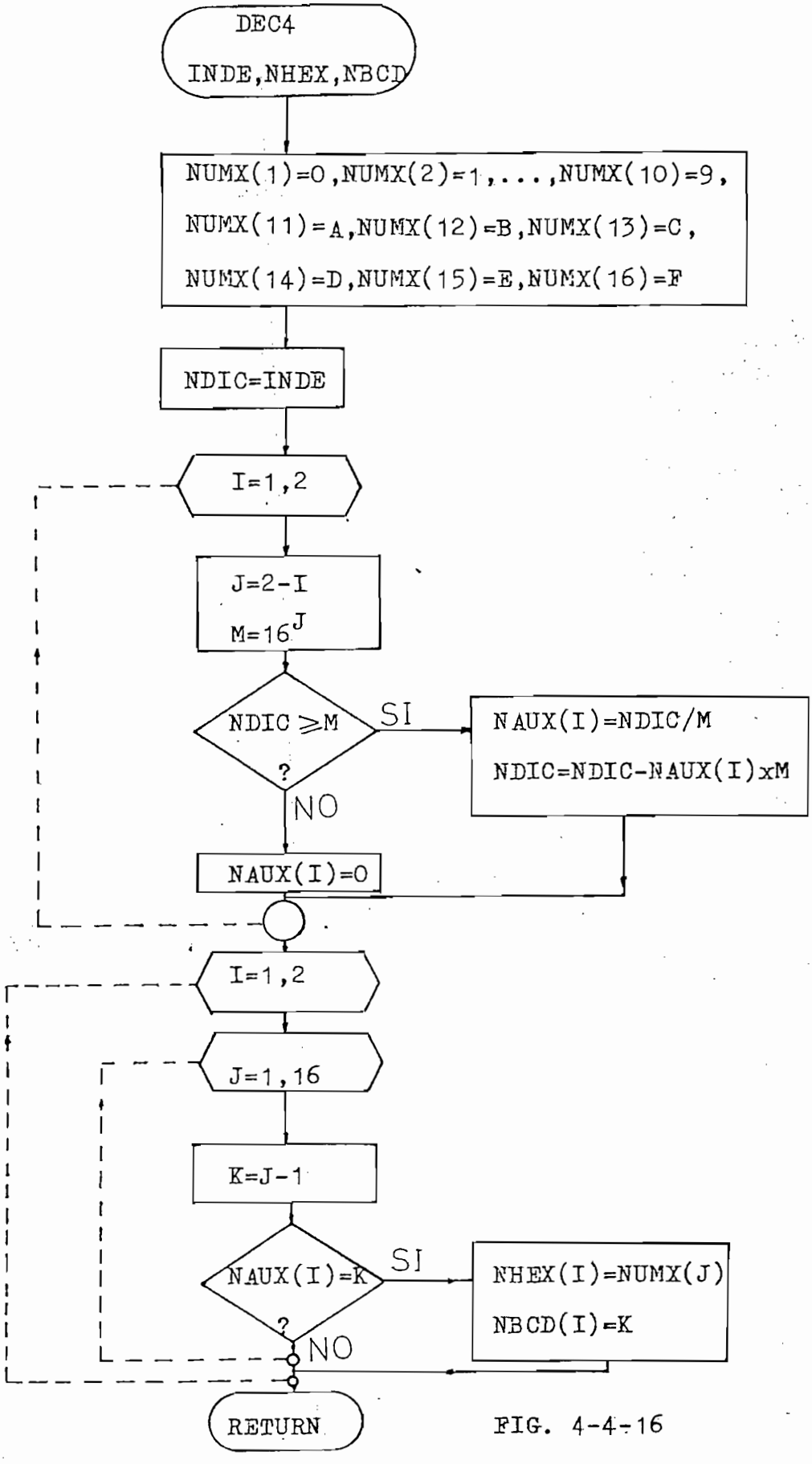


FIG. 4-4-16

SUBRUTINA LERR

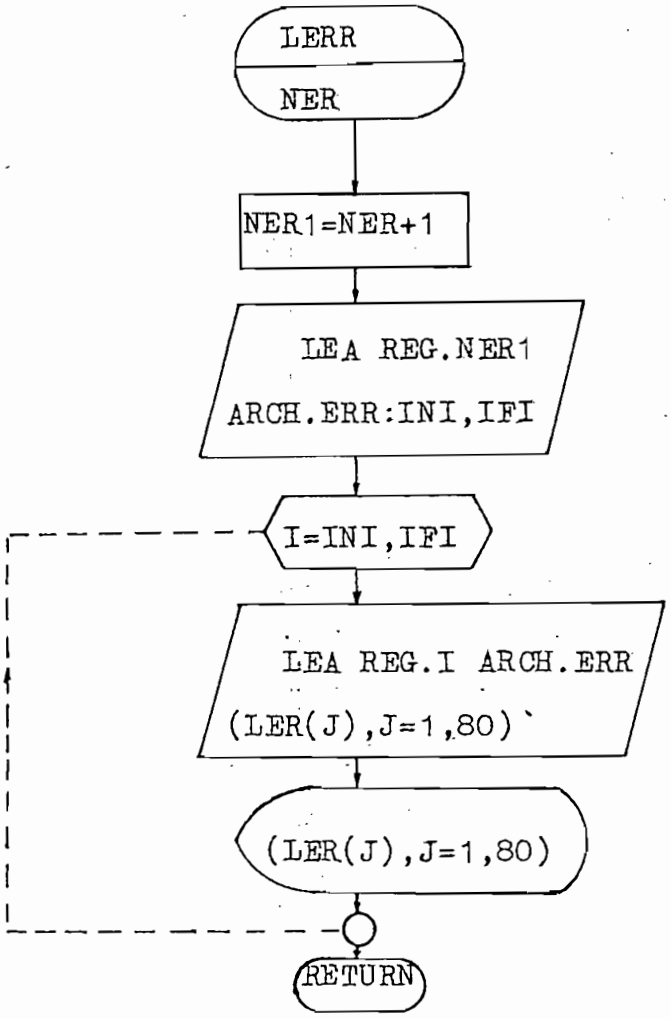


FIG. 4-4-17

SUBRUTINA ESPE

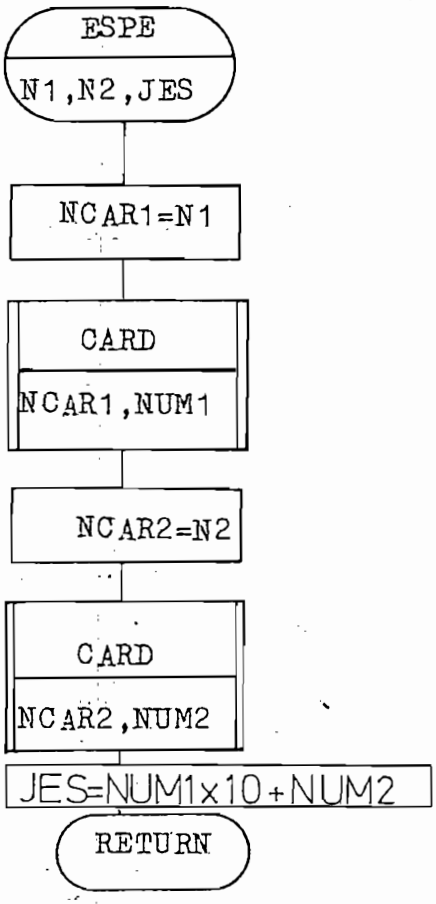


FIG. 4-4-18

SUBRUTINA DEK

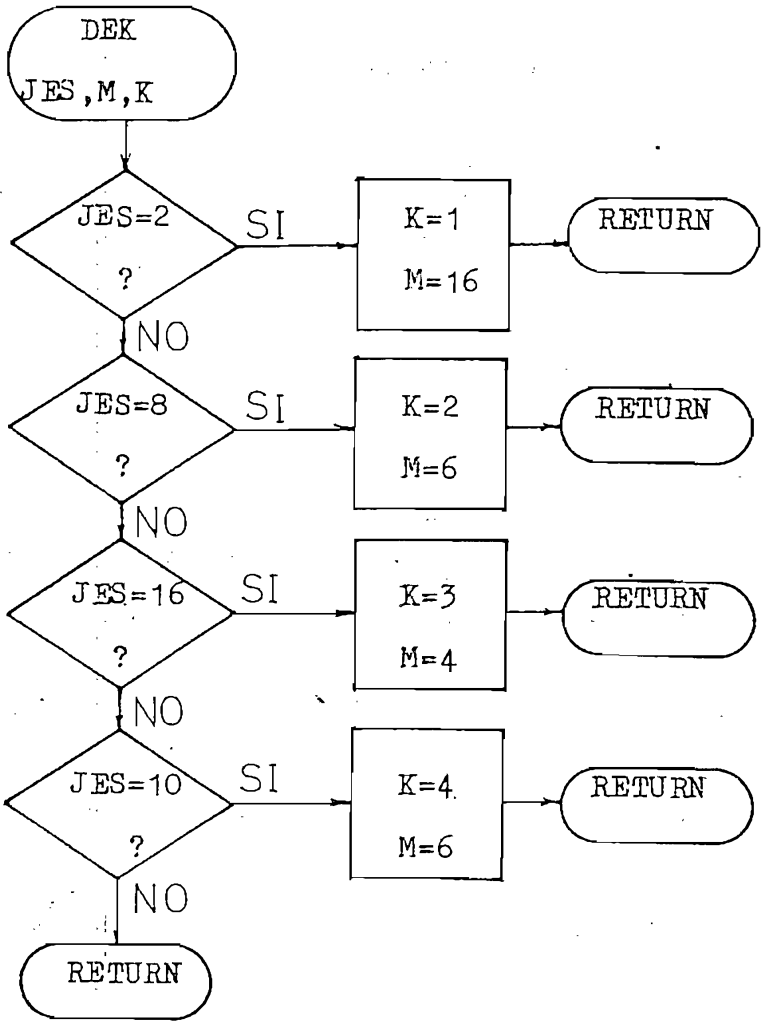


FIG. 4-4-19

SUBRUTINA RENUM

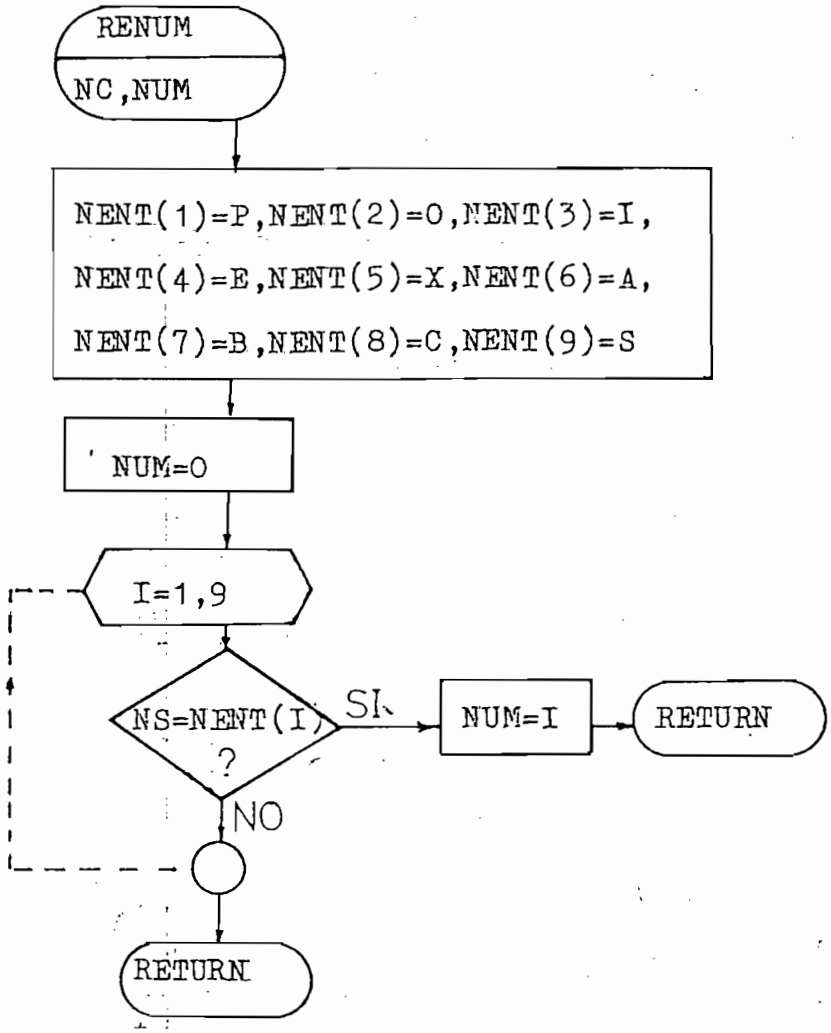


FIG. 4-4-20

SUBRUTINA DEBAS

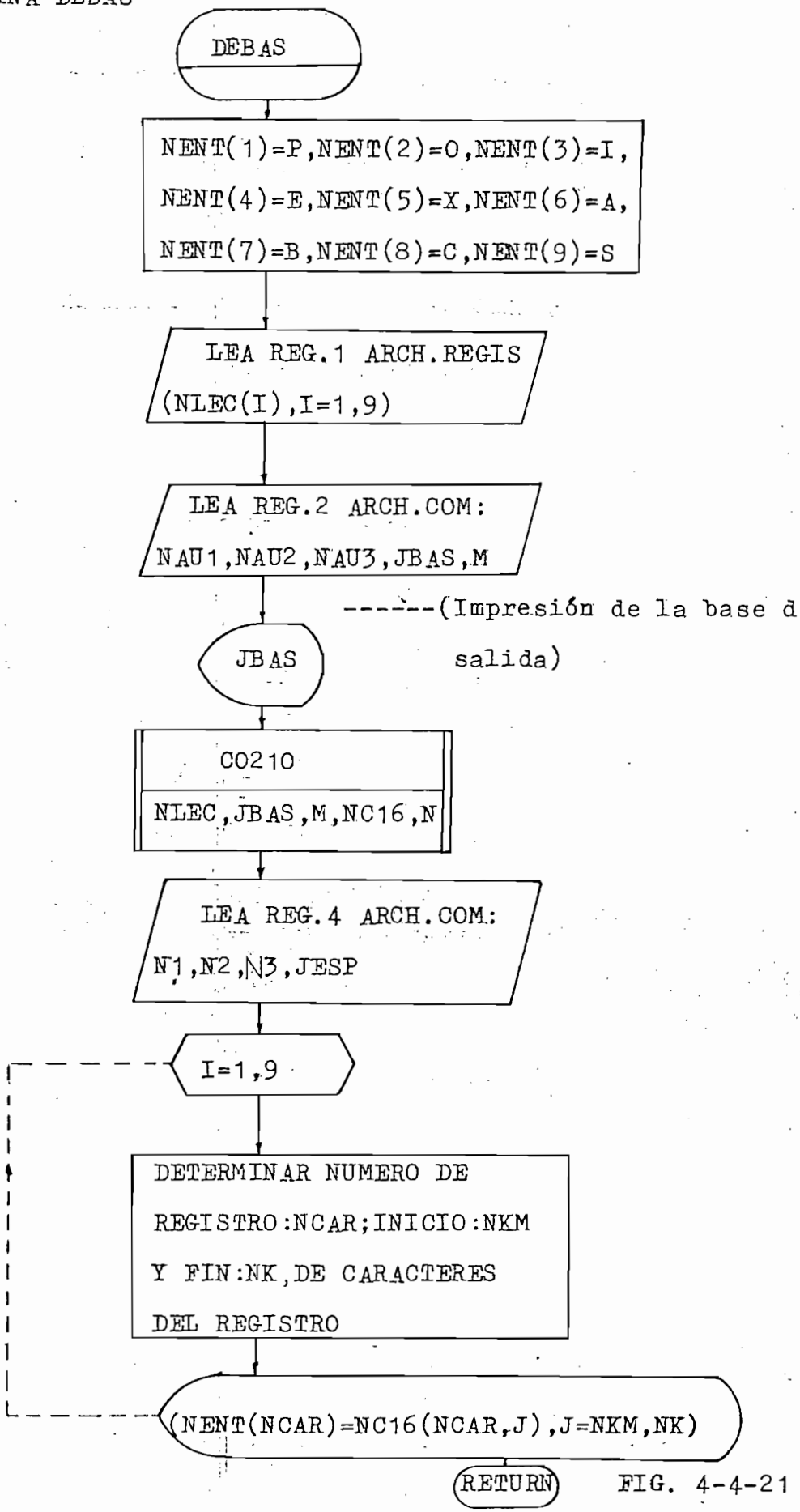


FIG. 4-4-21

SUBRUTINA REAO

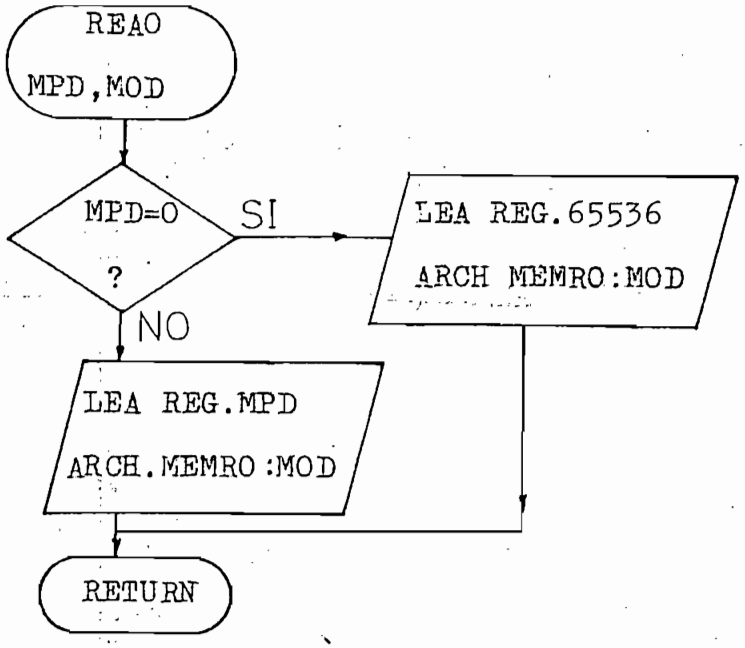


FIG. 4-4-22

SUBRUTINA DECE4

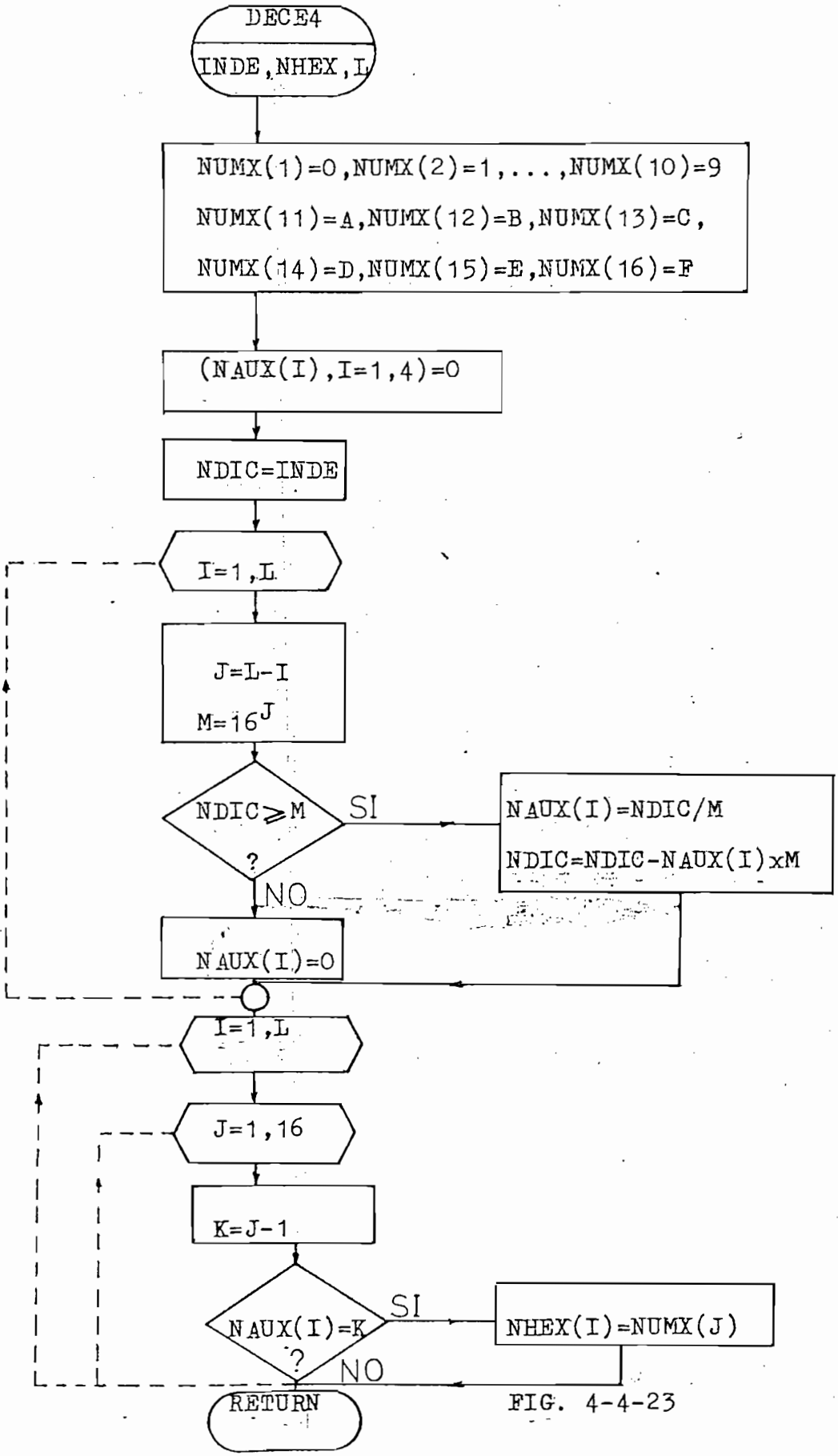


FIG. 4-4-23

SUBRUTINA DENT

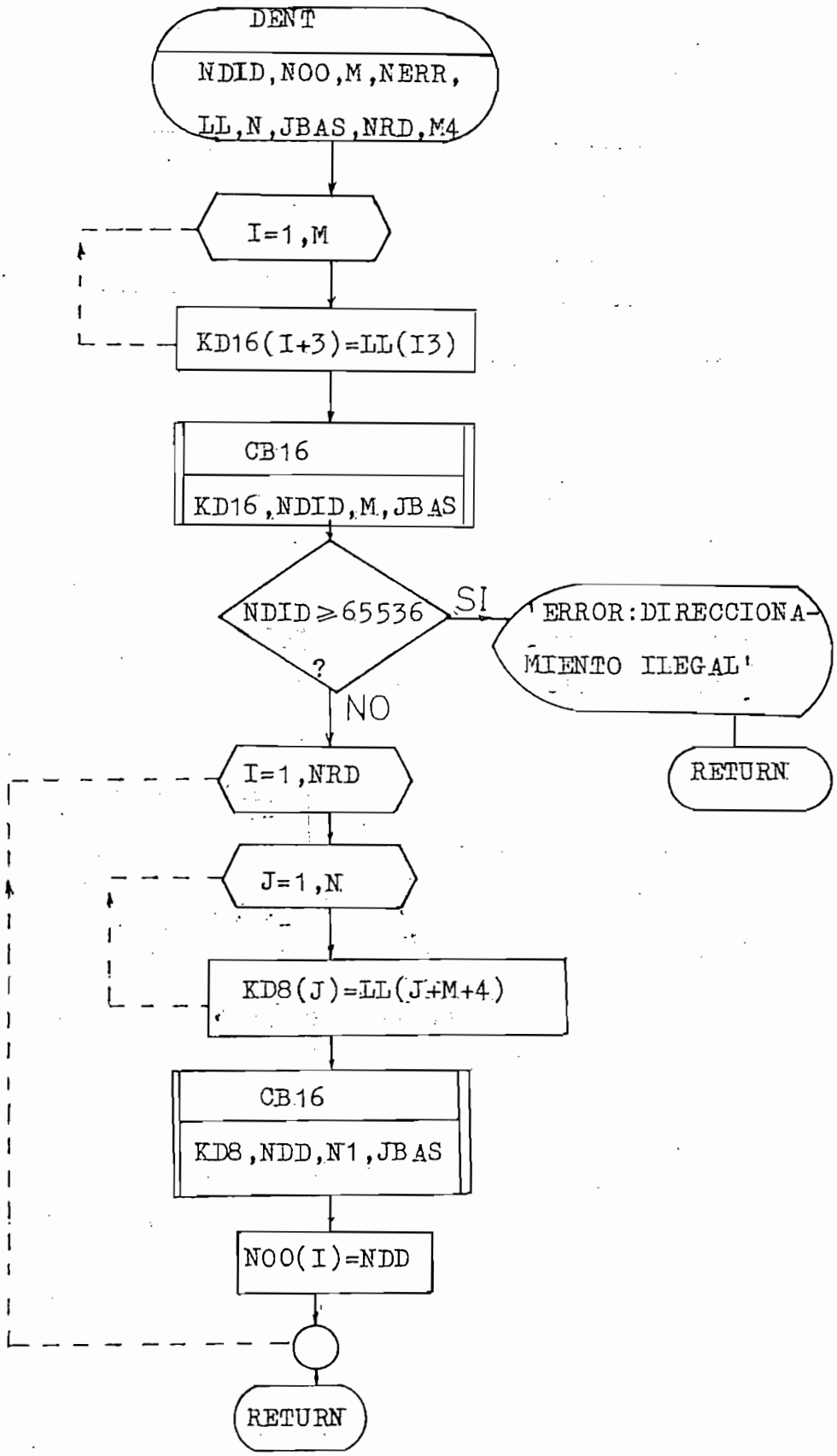


FIG. 4-4-24

SUBRUTINA IMPM

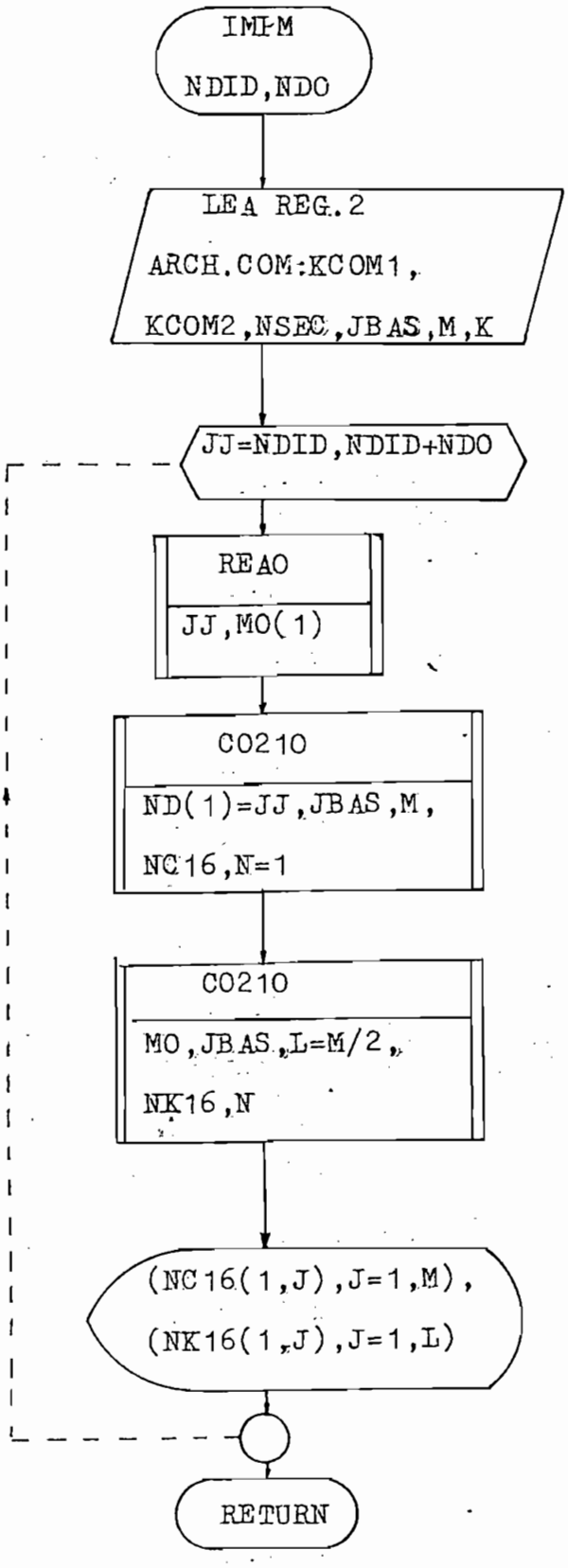
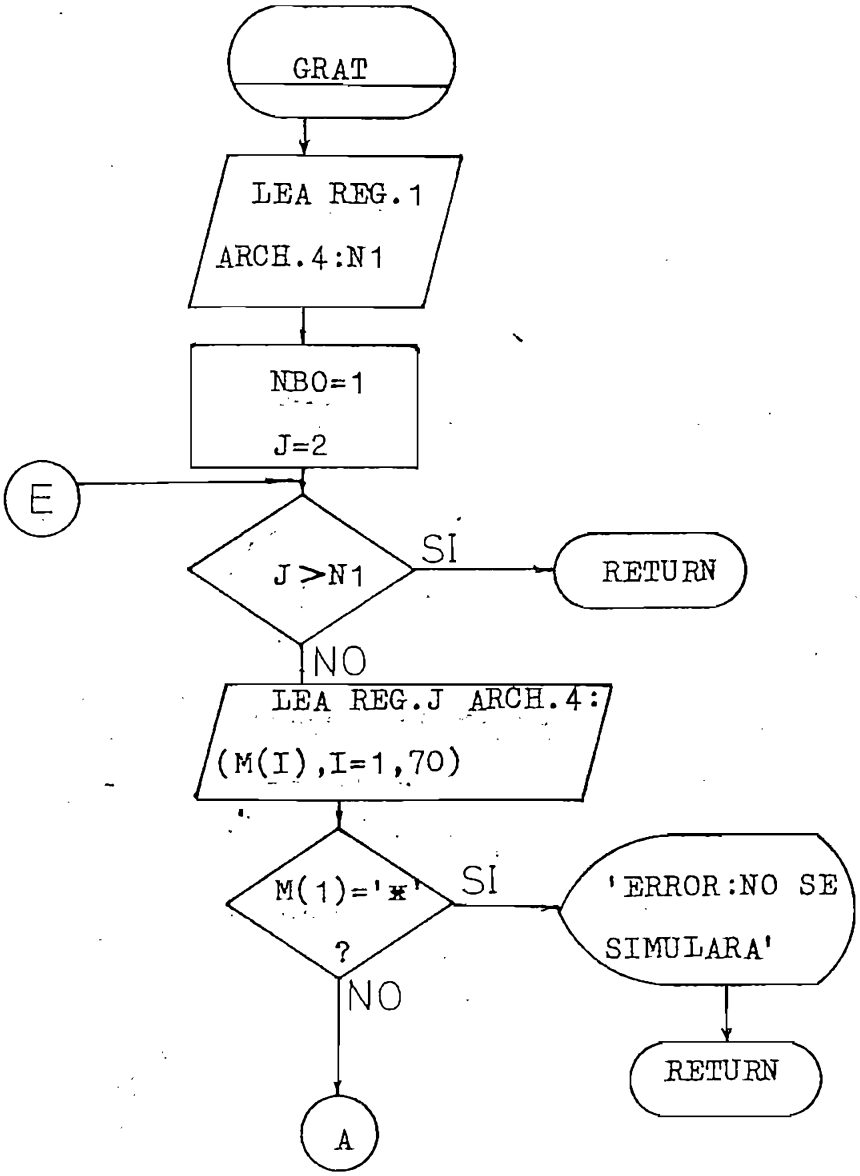


FIG. 4-4-25

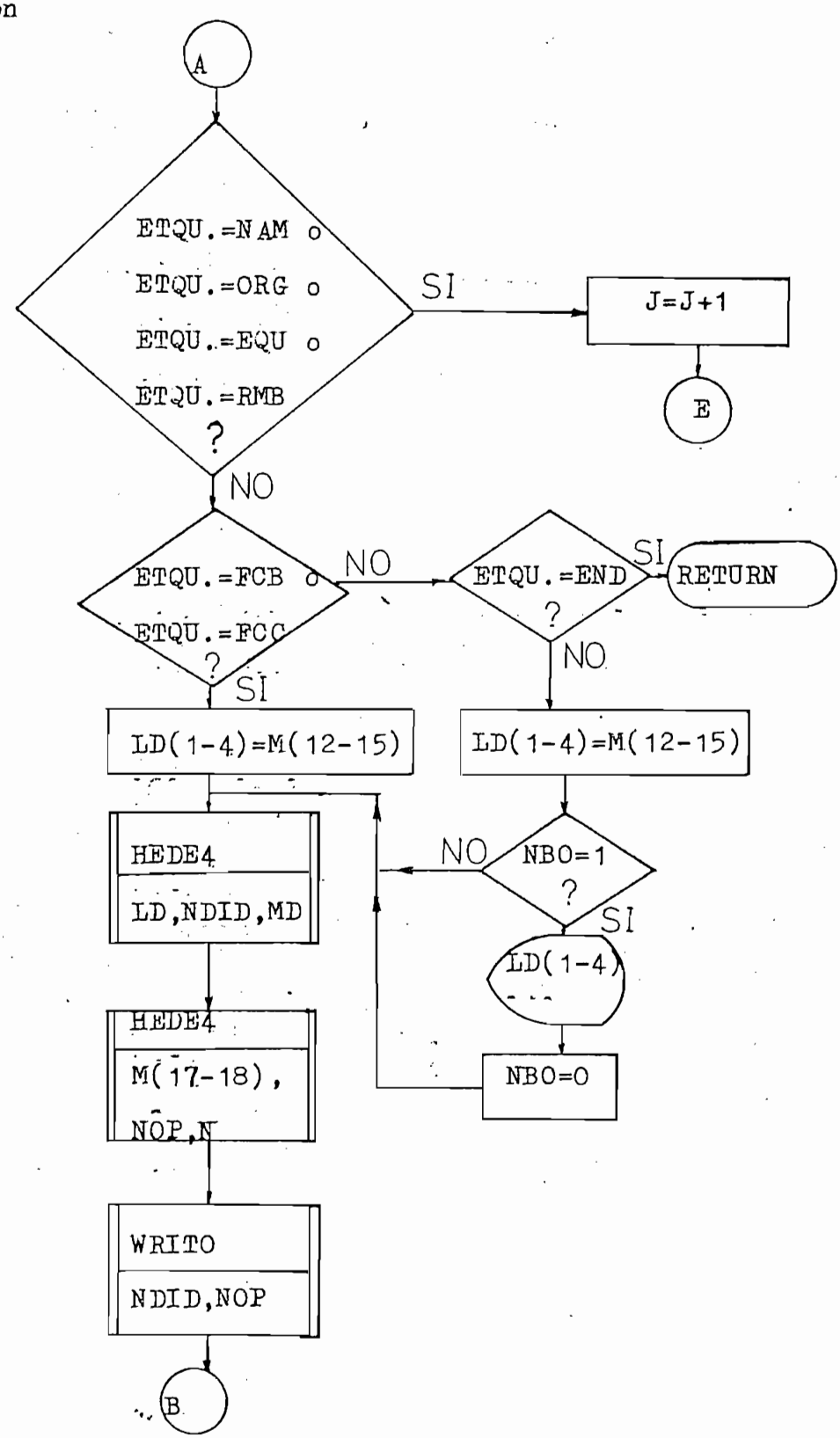
SUBRUTINA GRAT

Para realizar esta subrutina se toma en cuenta : la disposición del archivo que contiene el programa ensamblado en el Cross Assembler y la forma de utilizar las etiquetas del mismo.(Para referencia se tiene la Tesis del Cross-Assembler indicada en la Bibliografía).



SUBRUTINA GRAT

Continuación



SUBRUTINA GRAT

Continuación

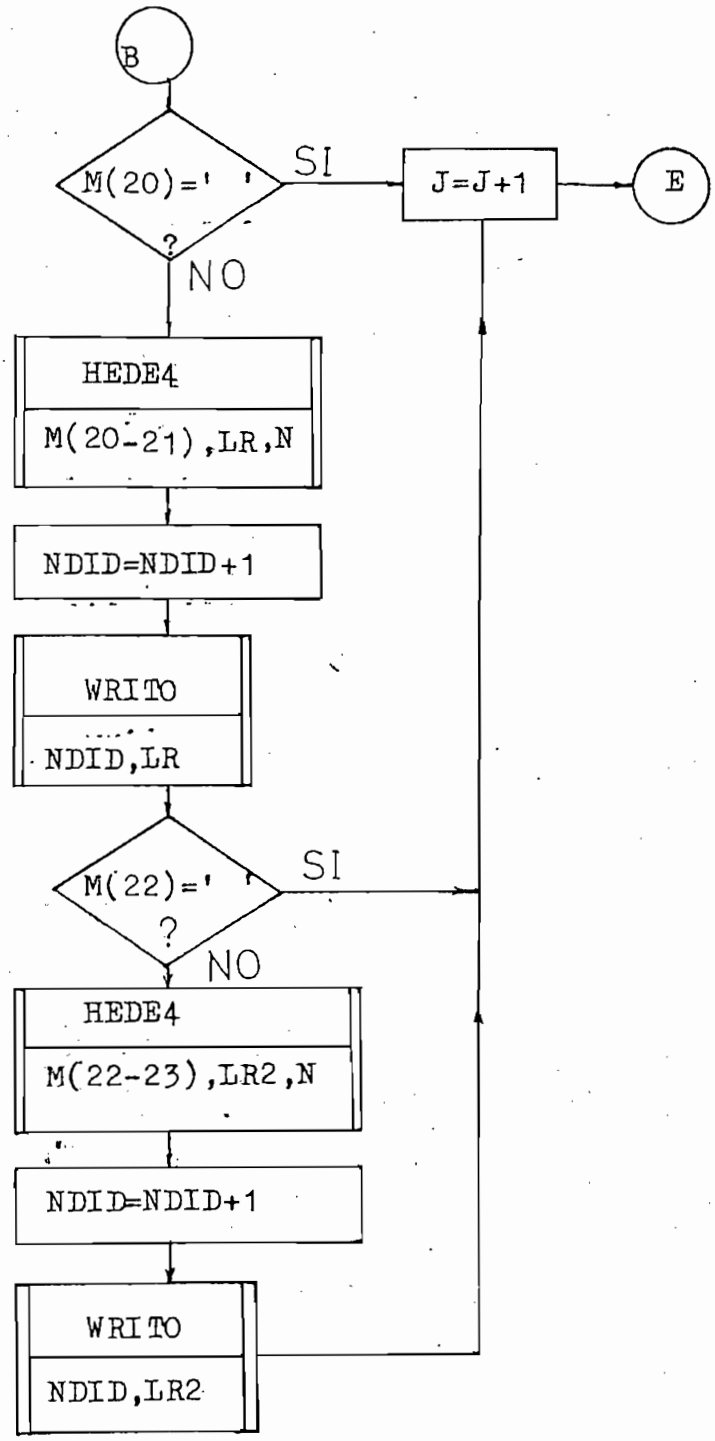
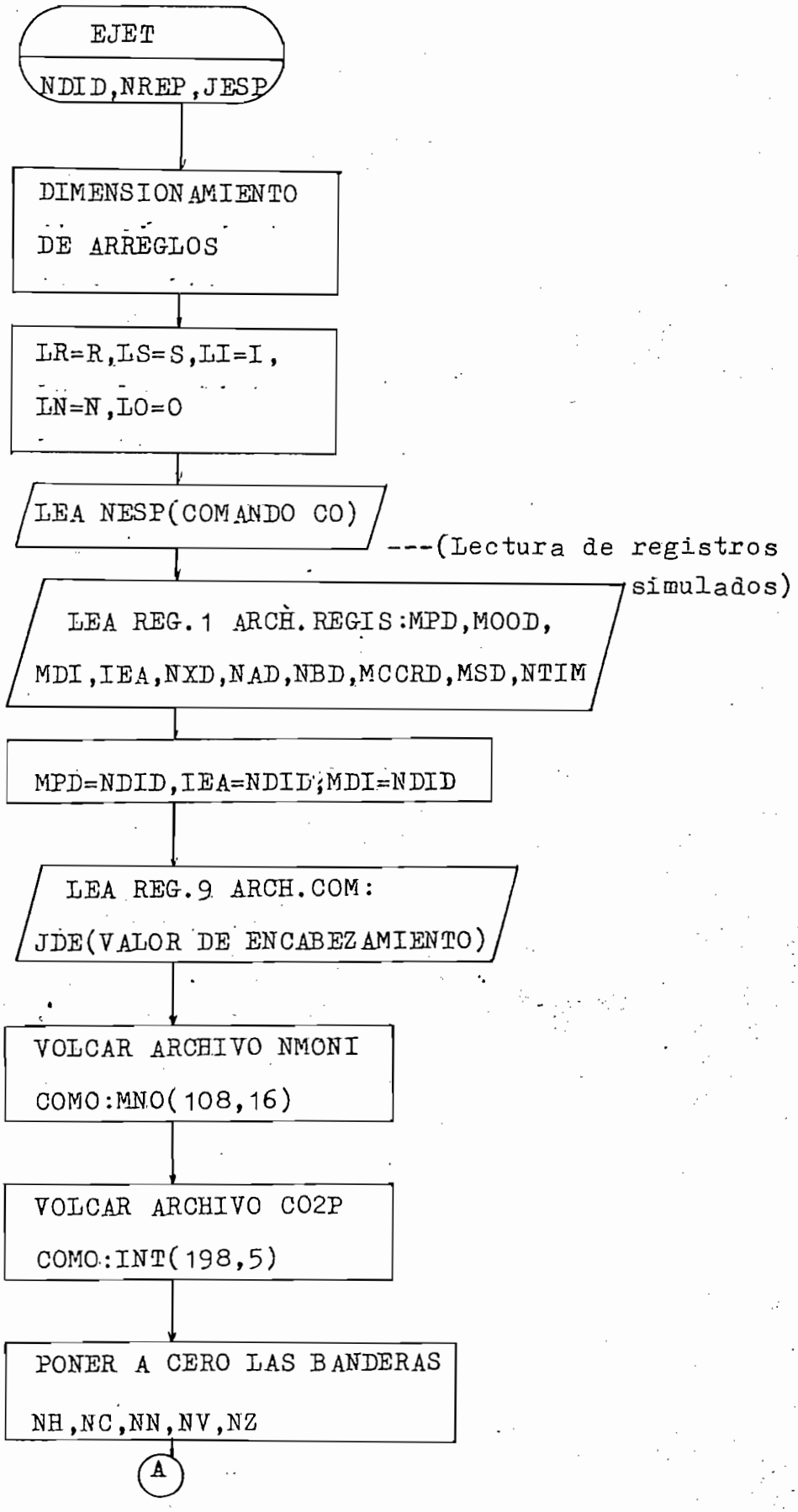


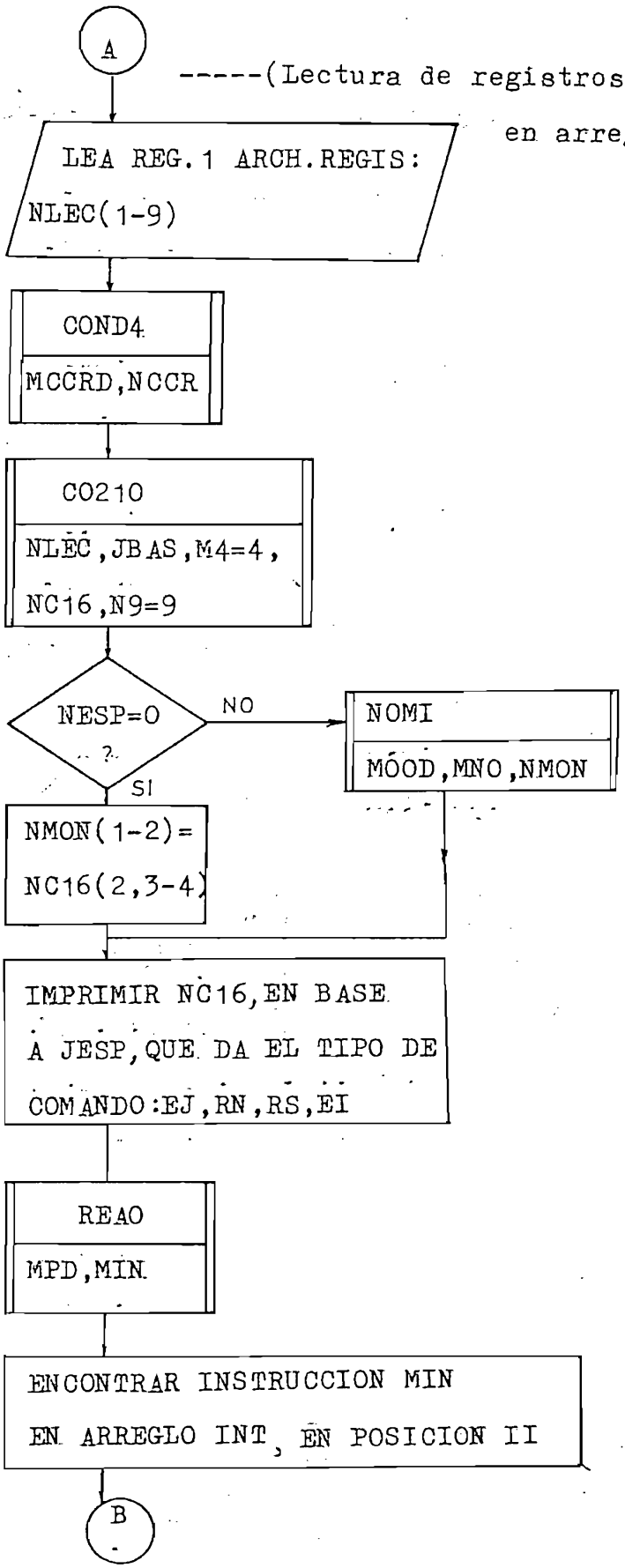
FIG. 4-4-26

SUBRUTINA EJET



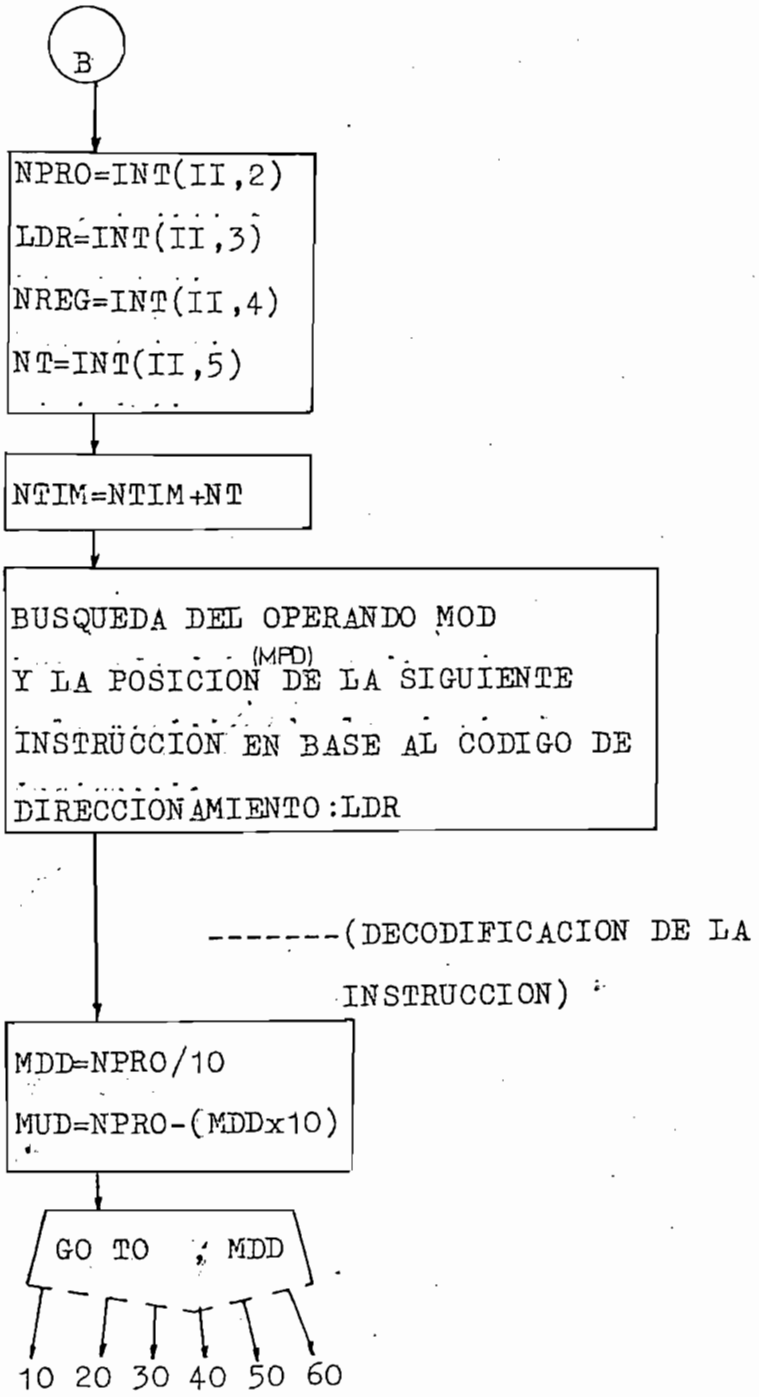
SUBRUTINA EJET

Continuación



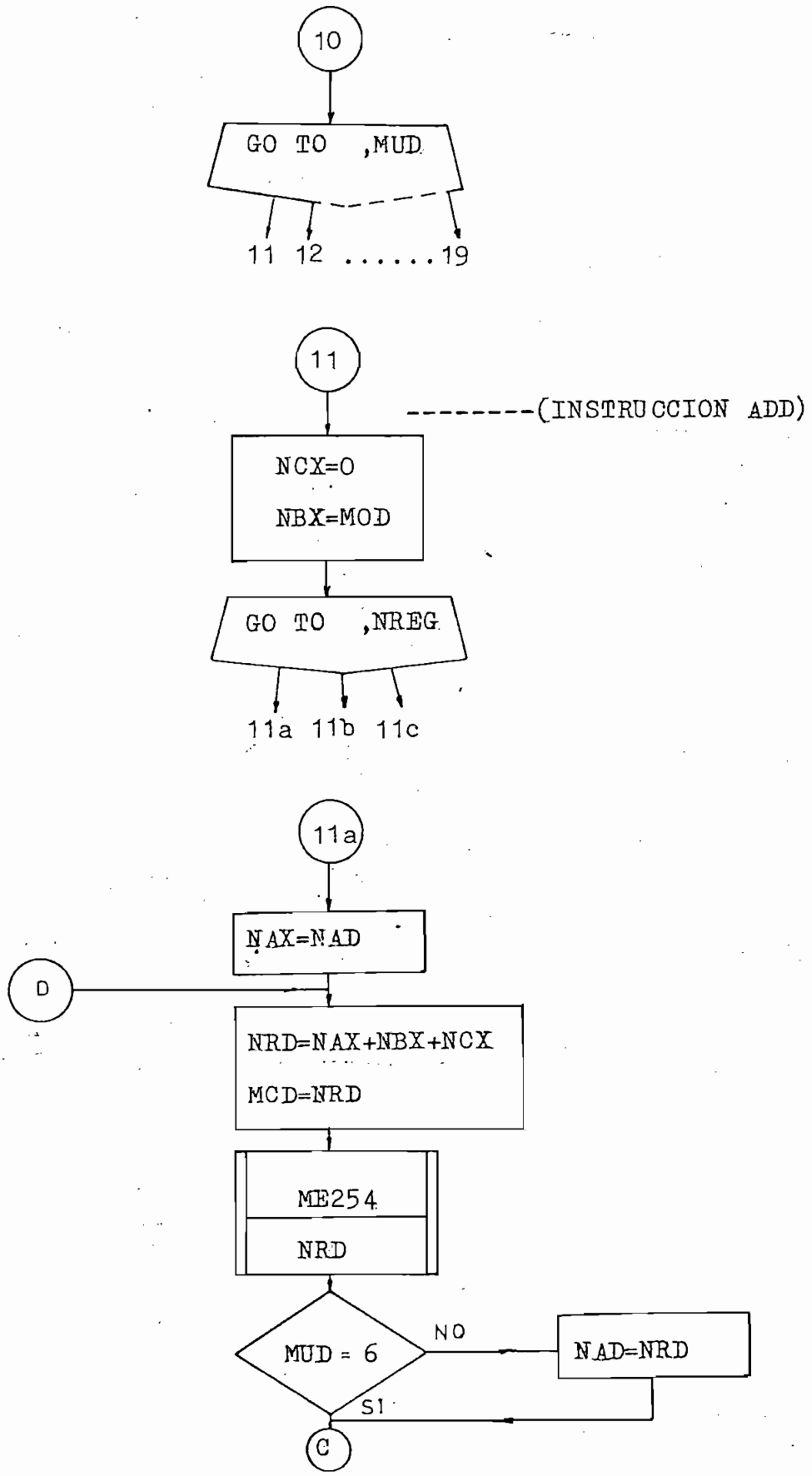
SUBRUTINA EJET

Continuación



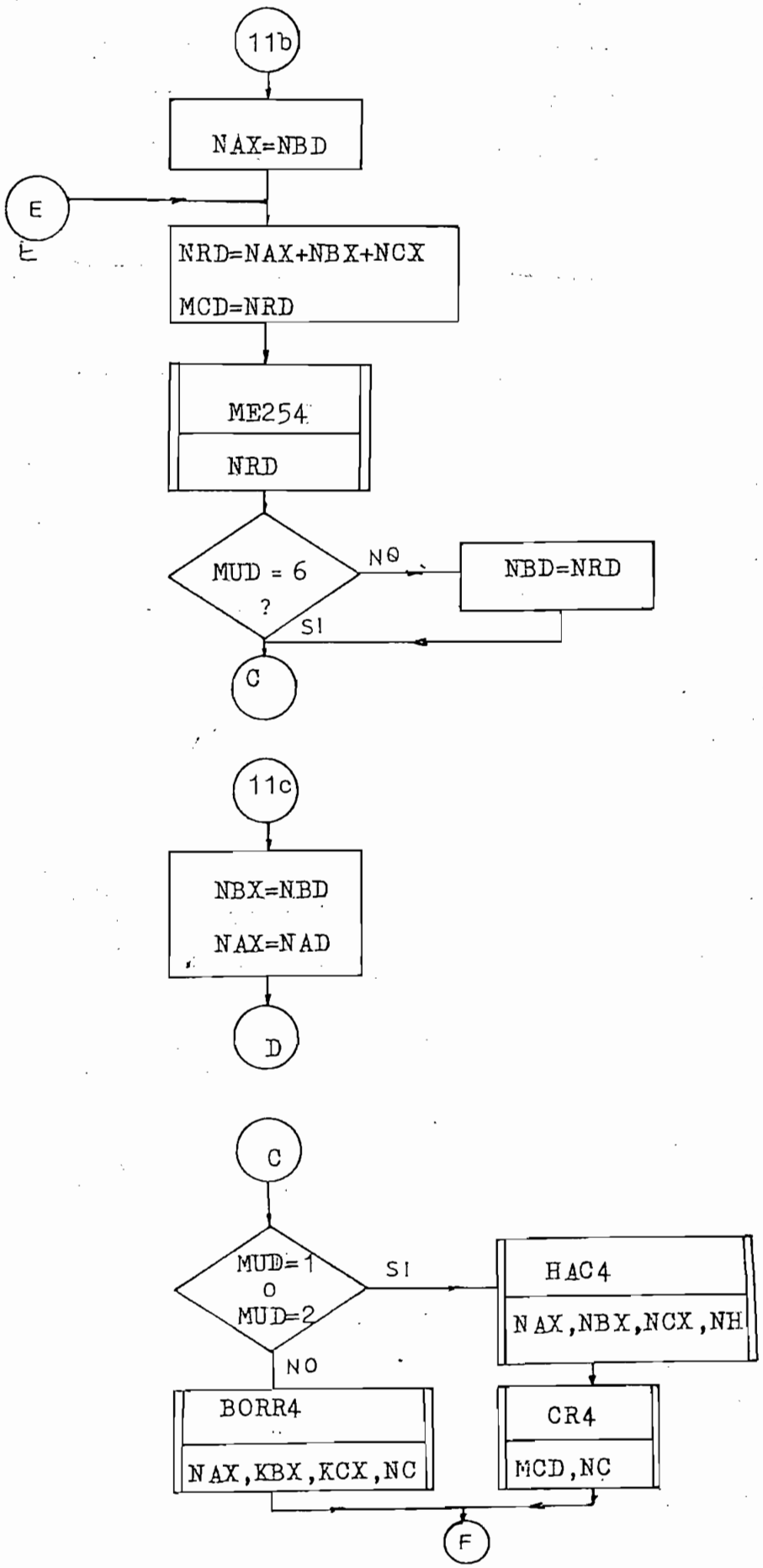
SUBRUTINA EJET

Continuación



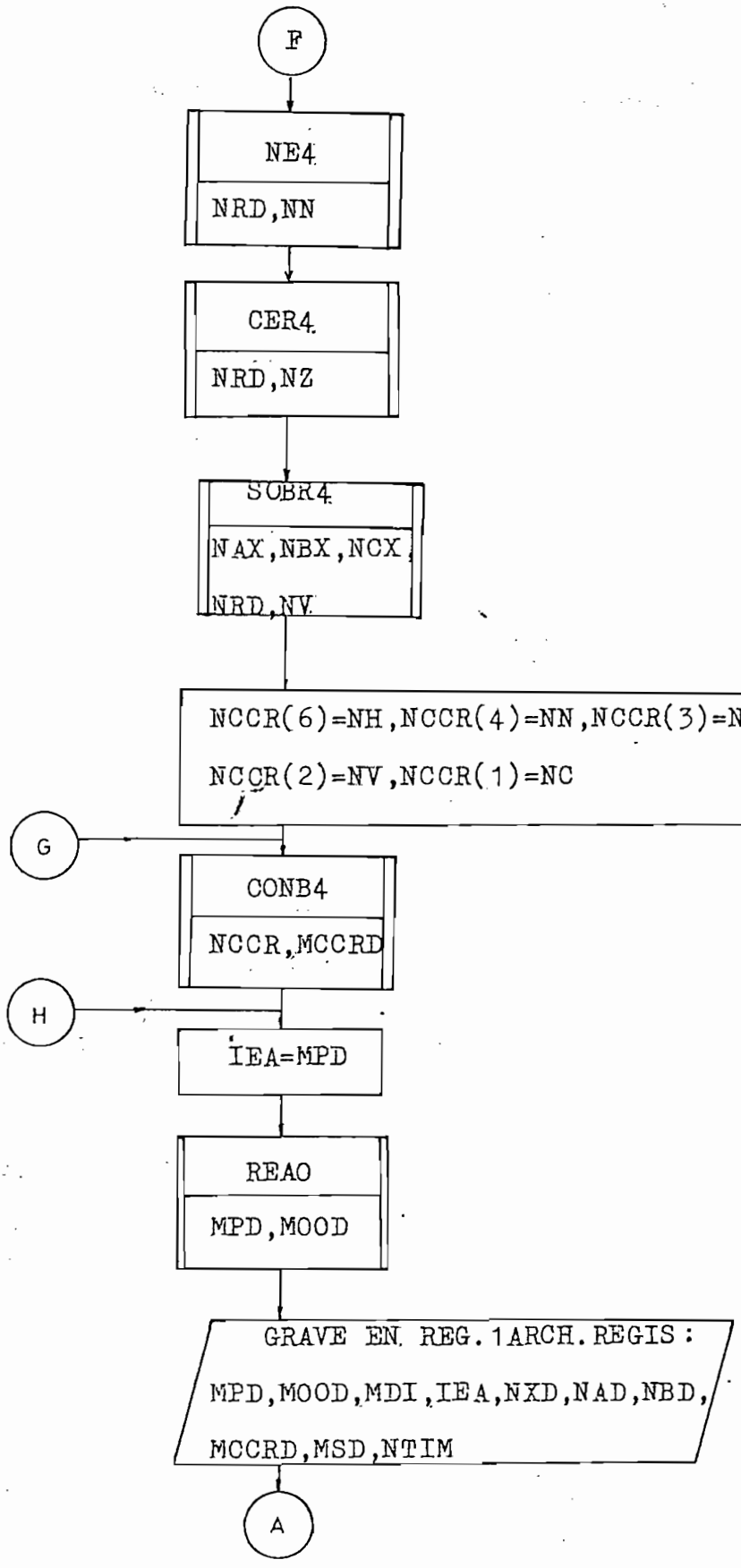
SUBRUTINA EJET

Continuación



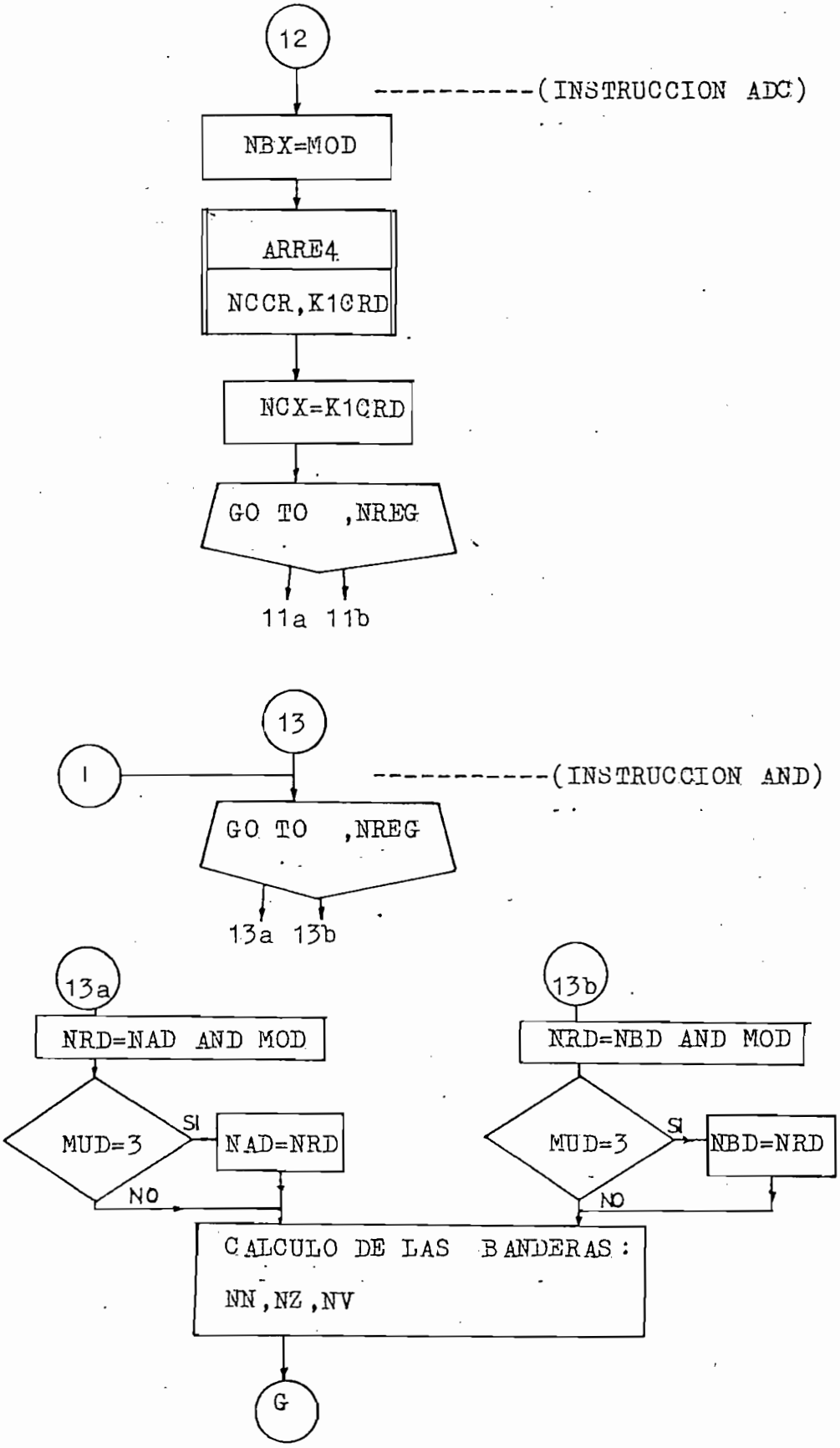
SUBRUTINA EJET

Continuación



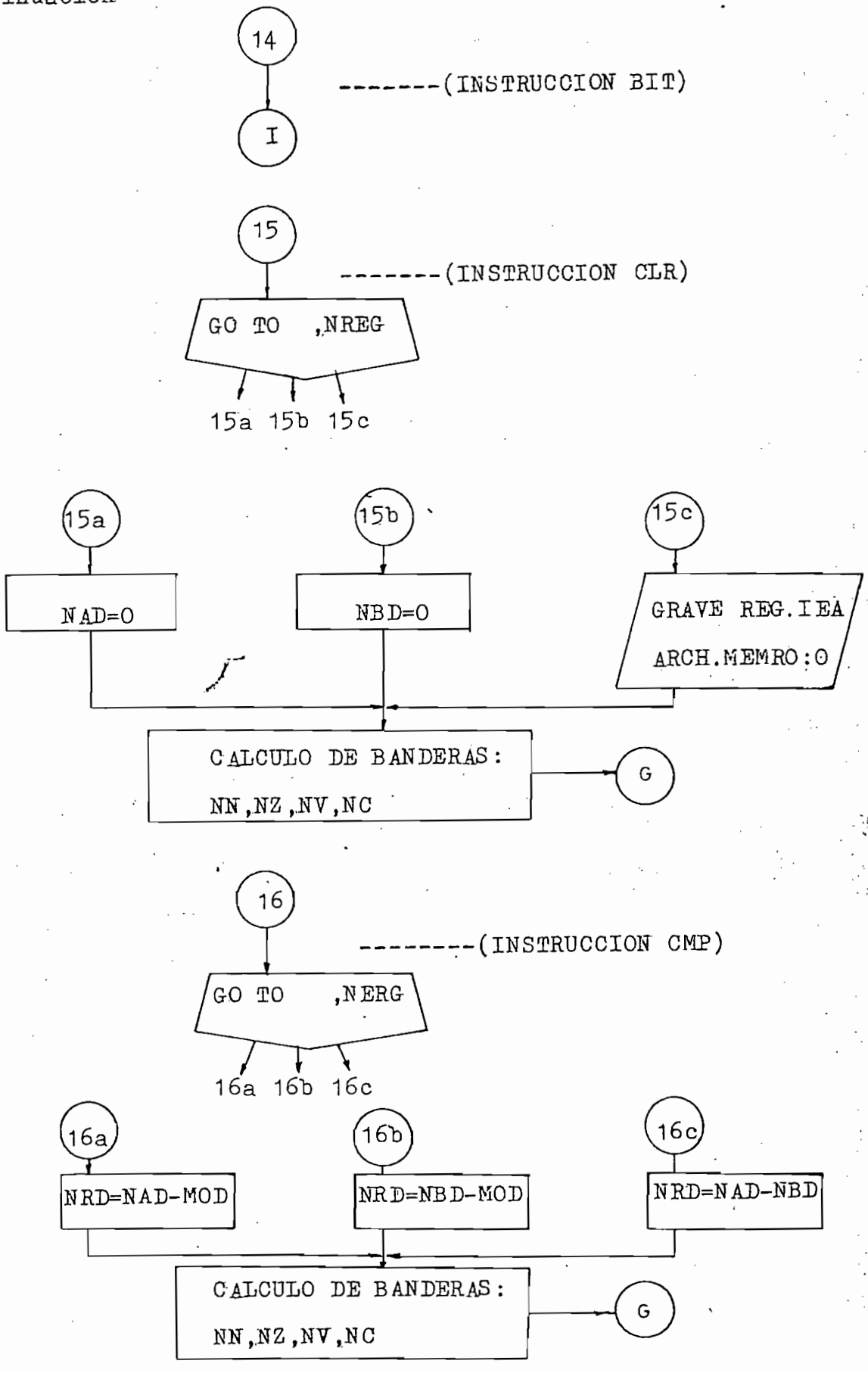
SUBRUTINA EJET

Continuación



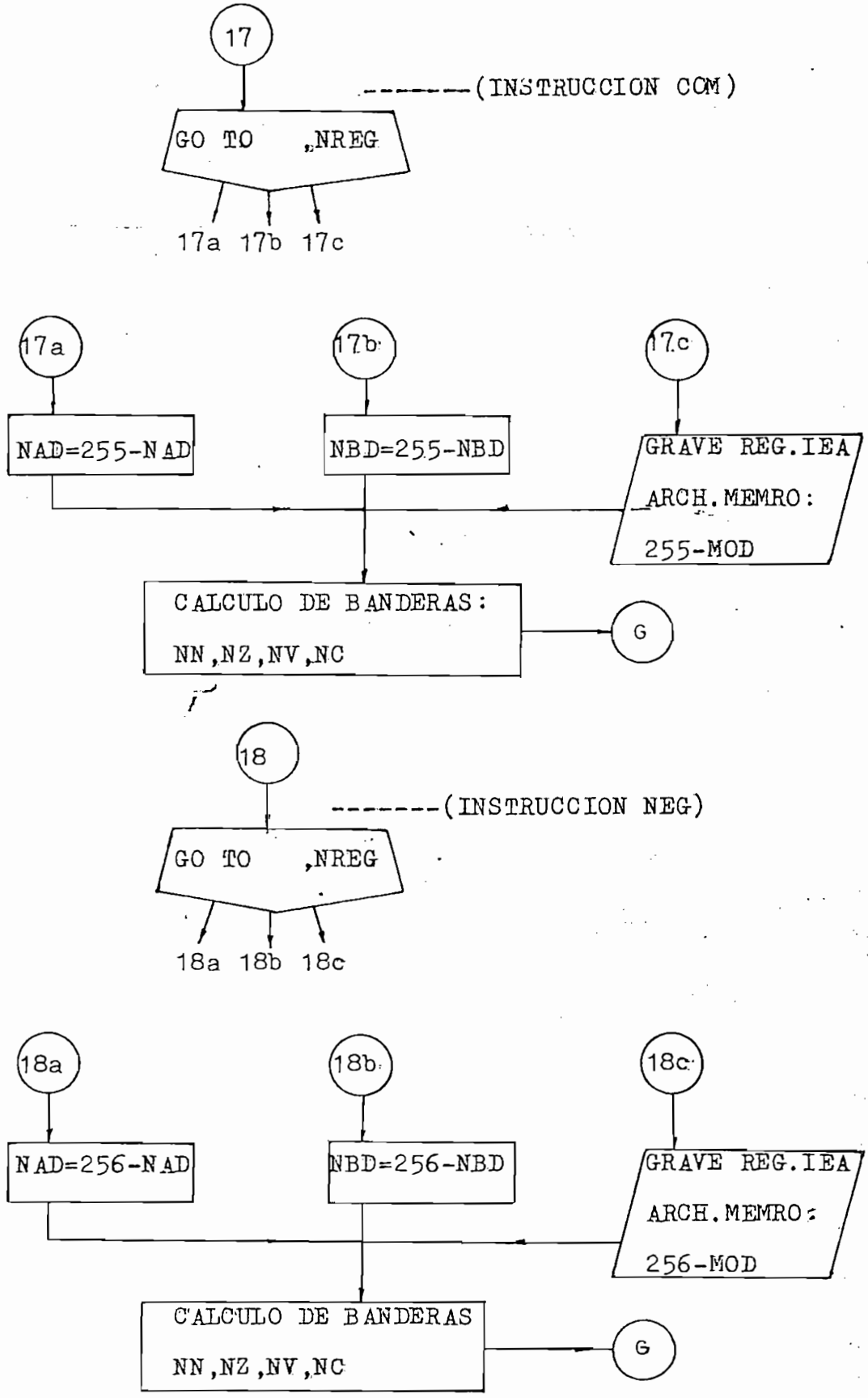
SUBRUTINA EJET

Continuación



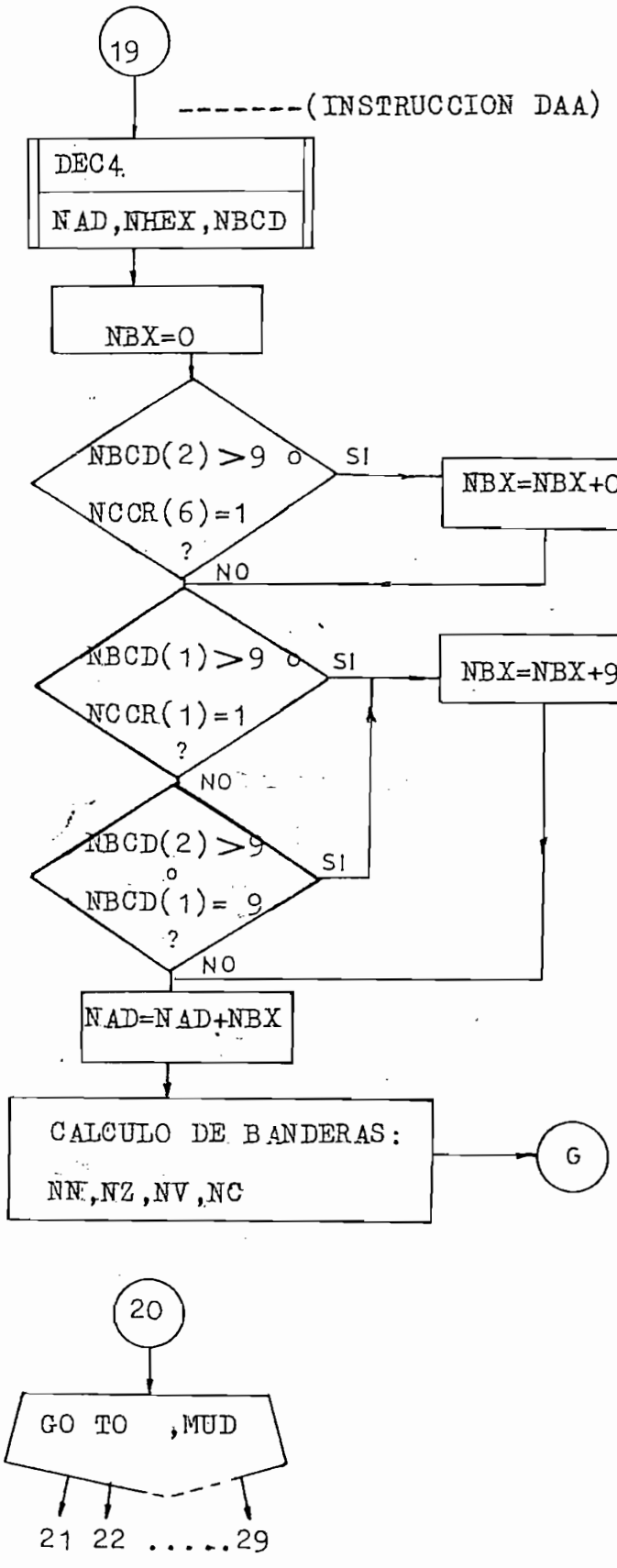
SUBRUTINA EJET

Continuación



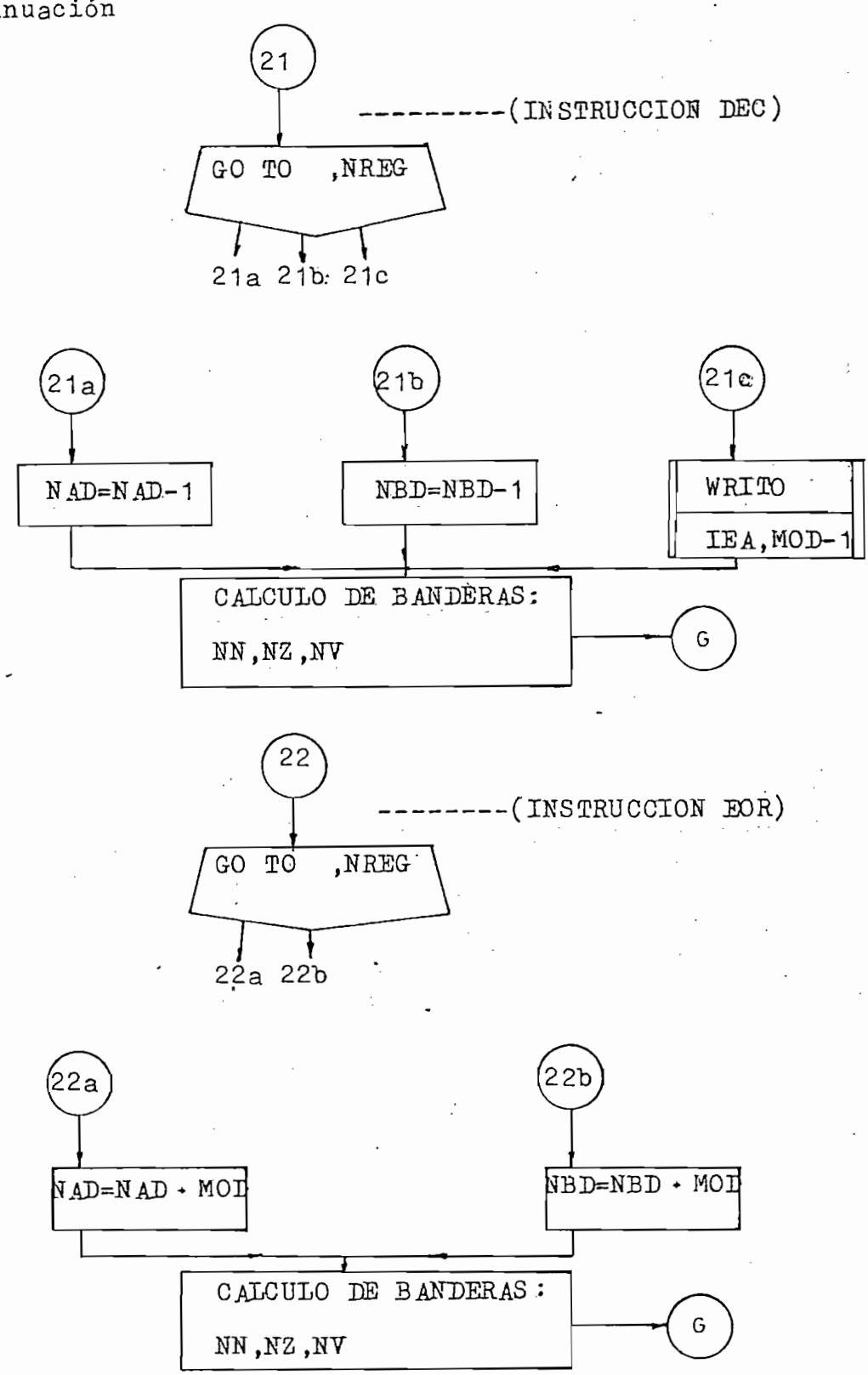
SUBRUTINA EJET

Continuación



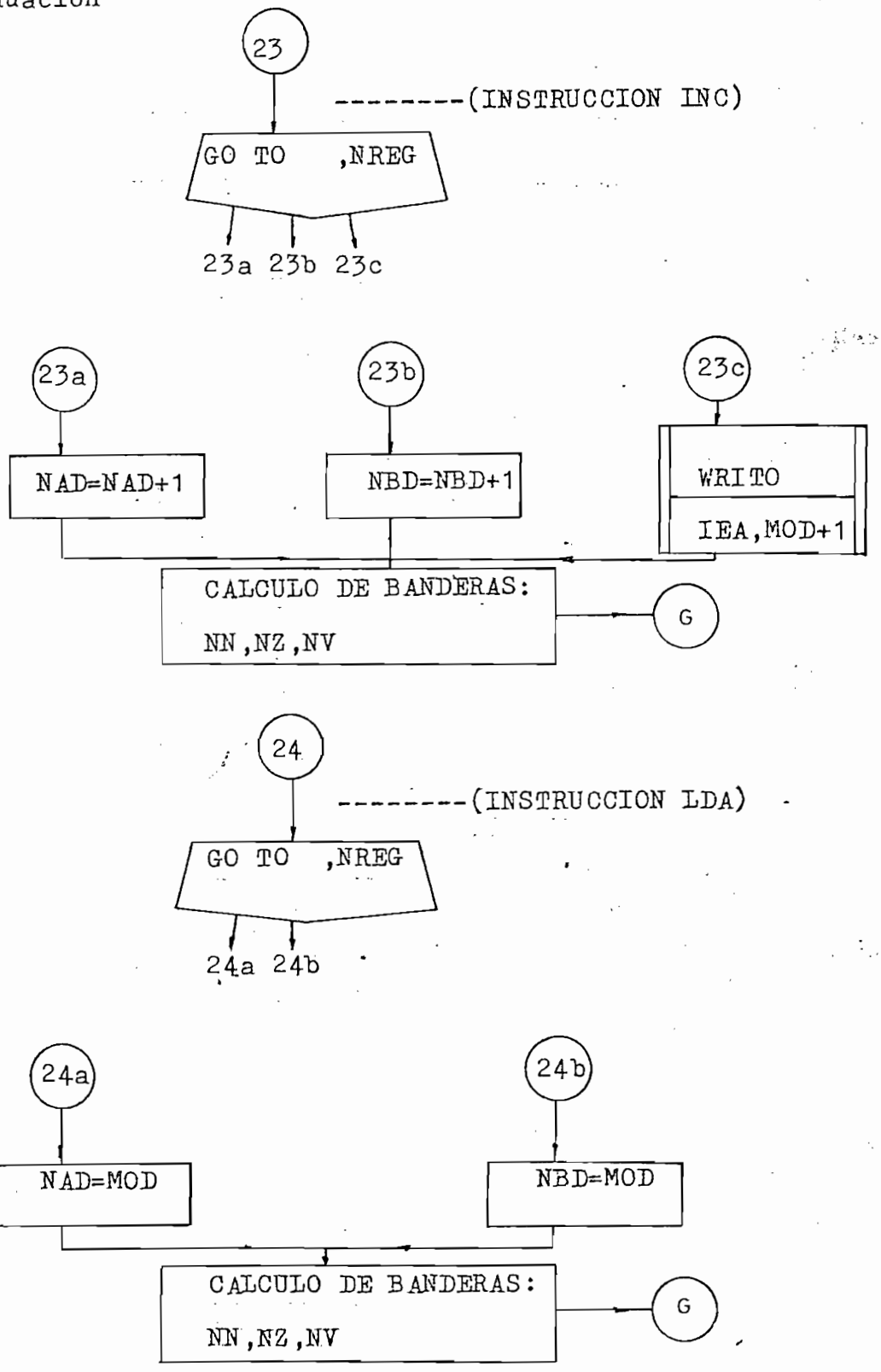
SUBRUTINA EJET

Continuación



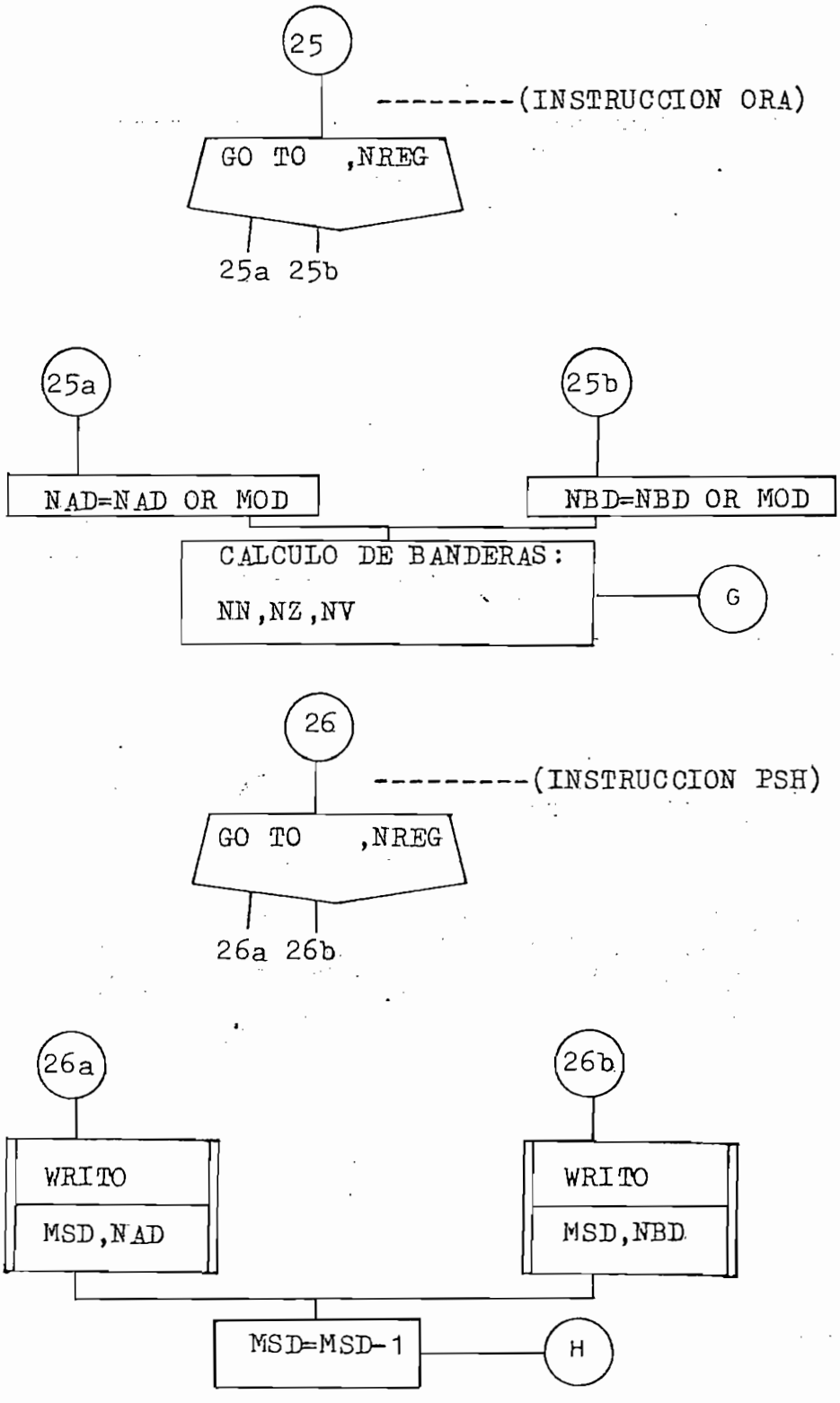
SUBRUTINA EJET

Continuación



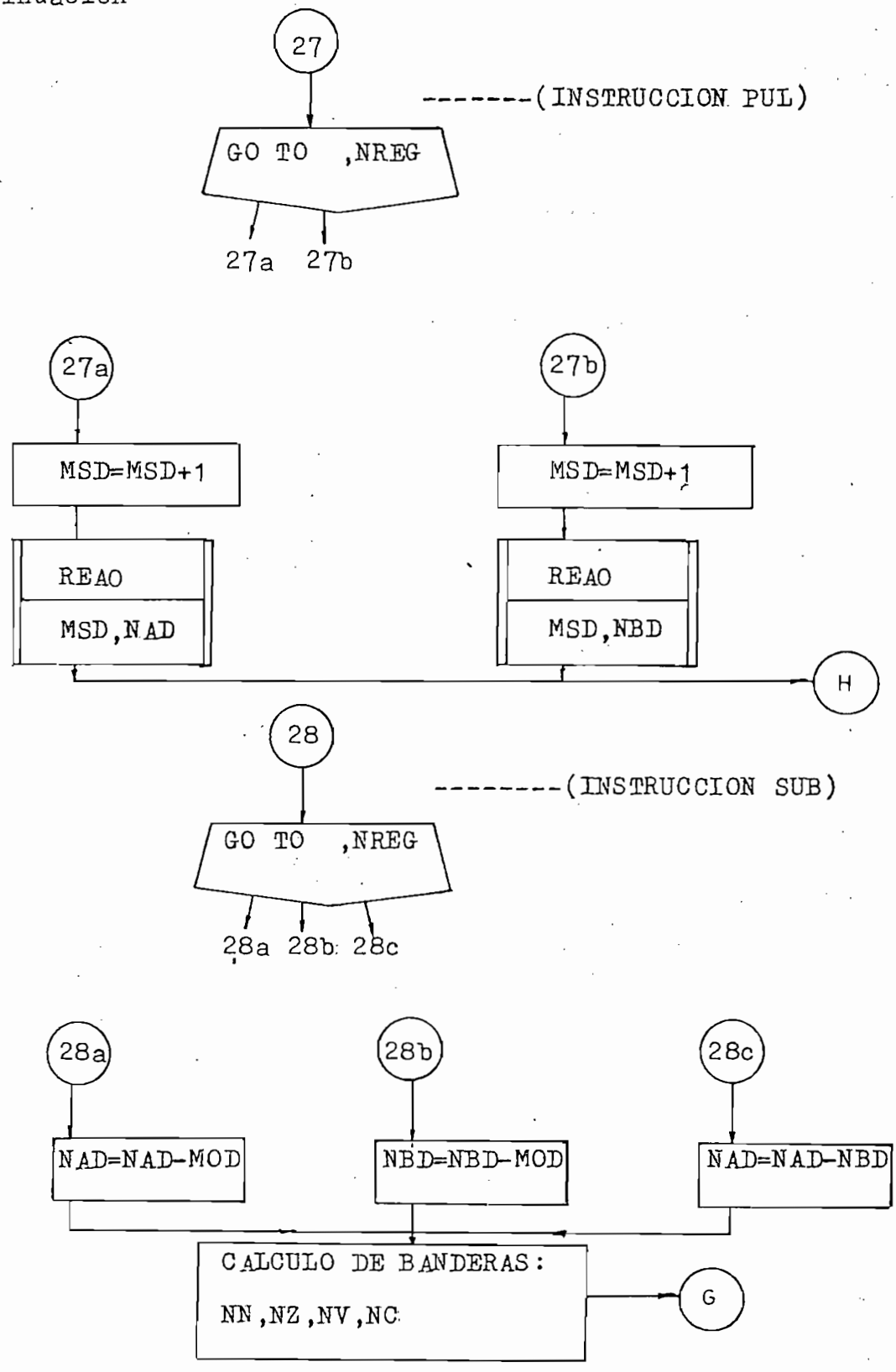
SUBRUTINA EJET

Continuación



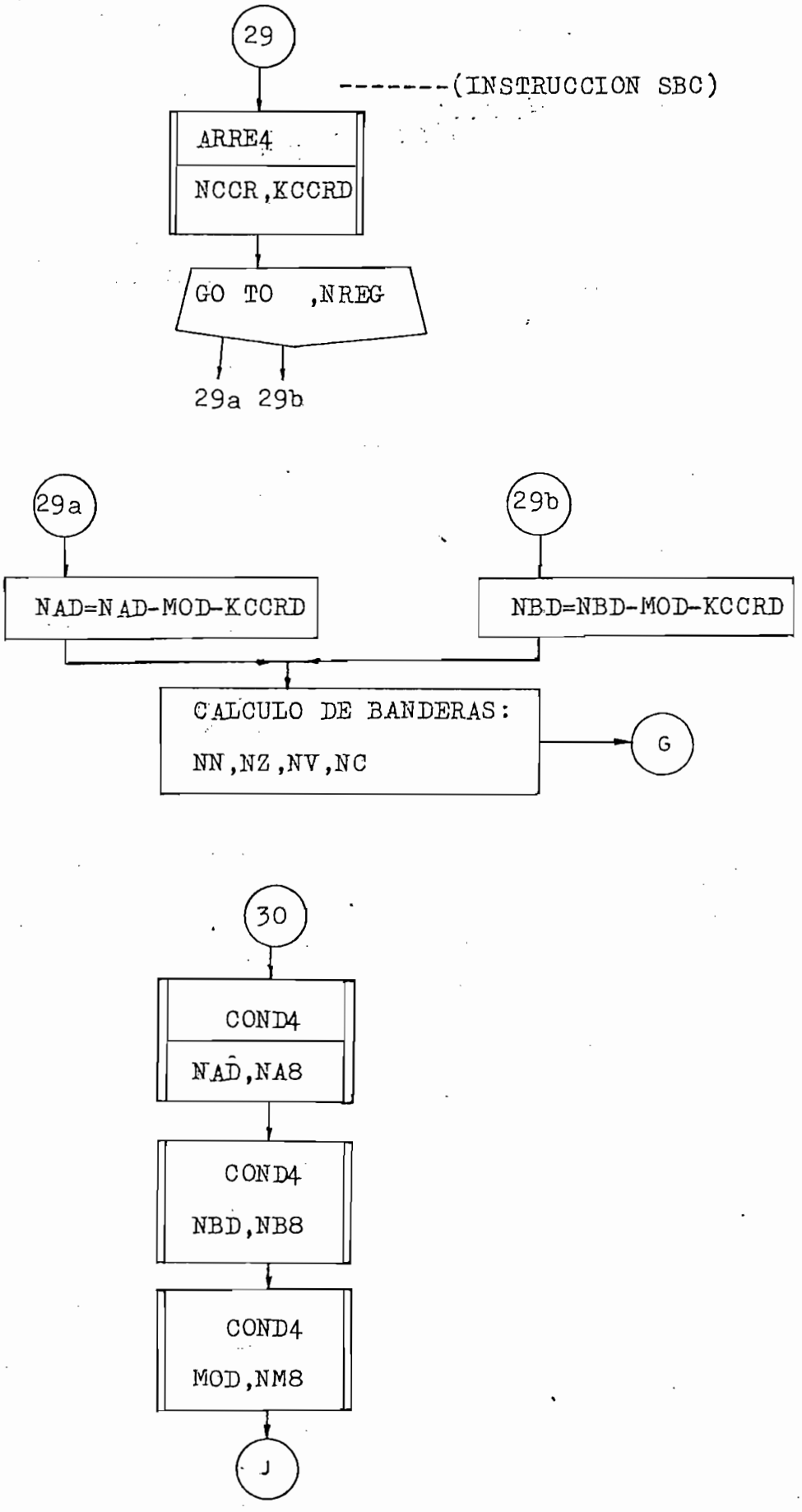
SUBRUTINA EJET

Continuación



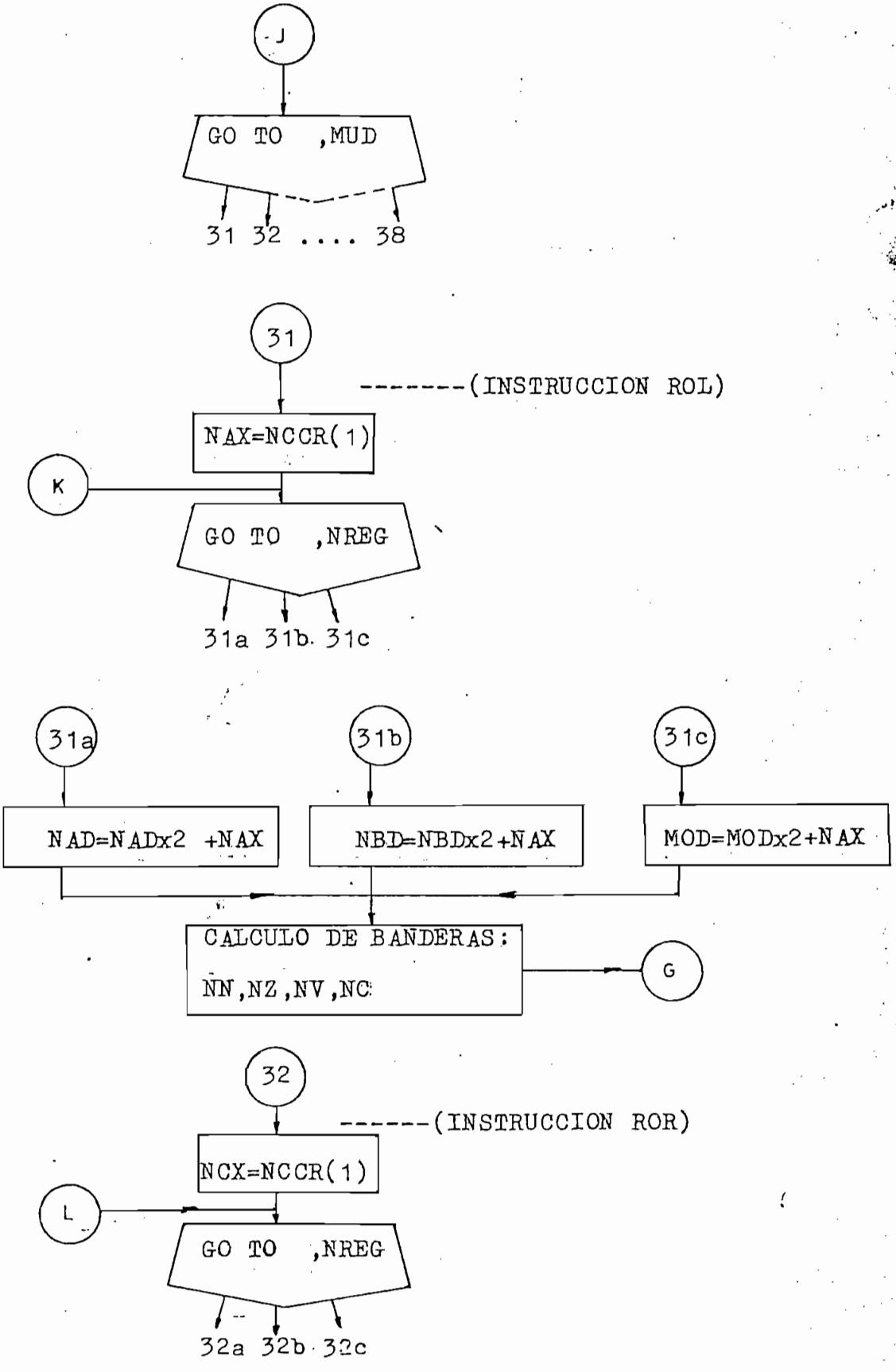
SUBRUTINA EJET

Continuación



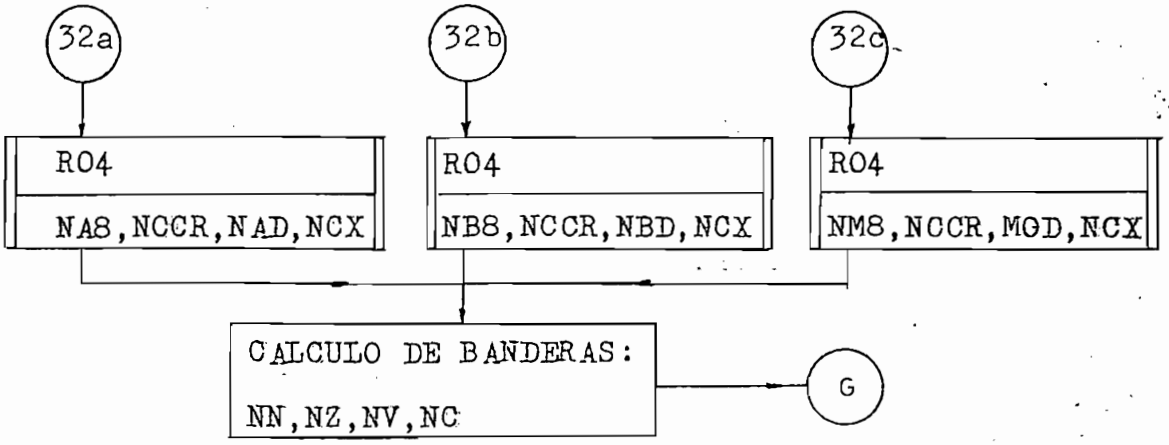
SUBRUTINA EJET

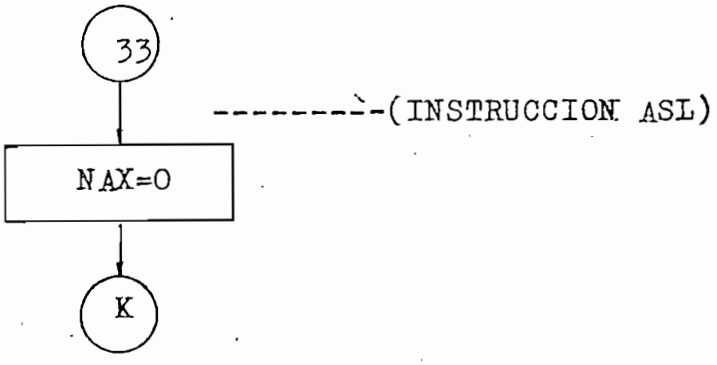
Continuación

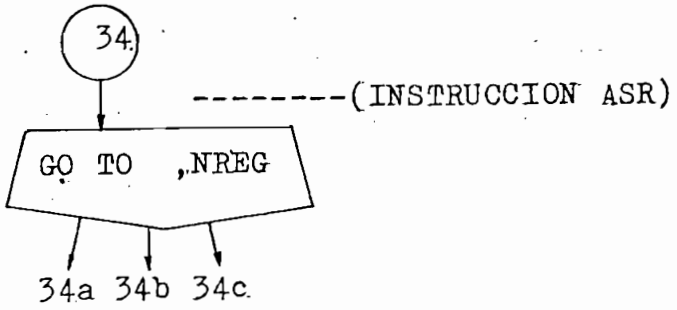


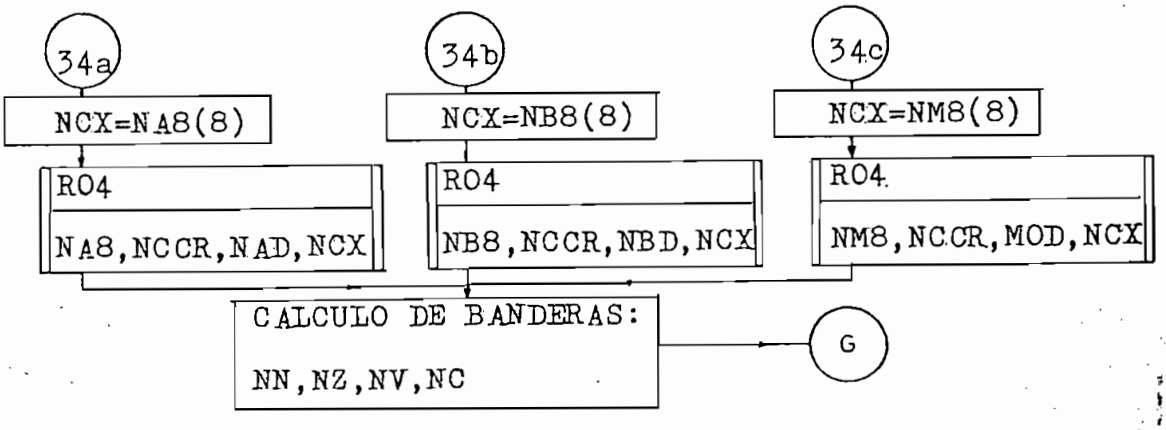
SUBRUTINA EJET

Continuación



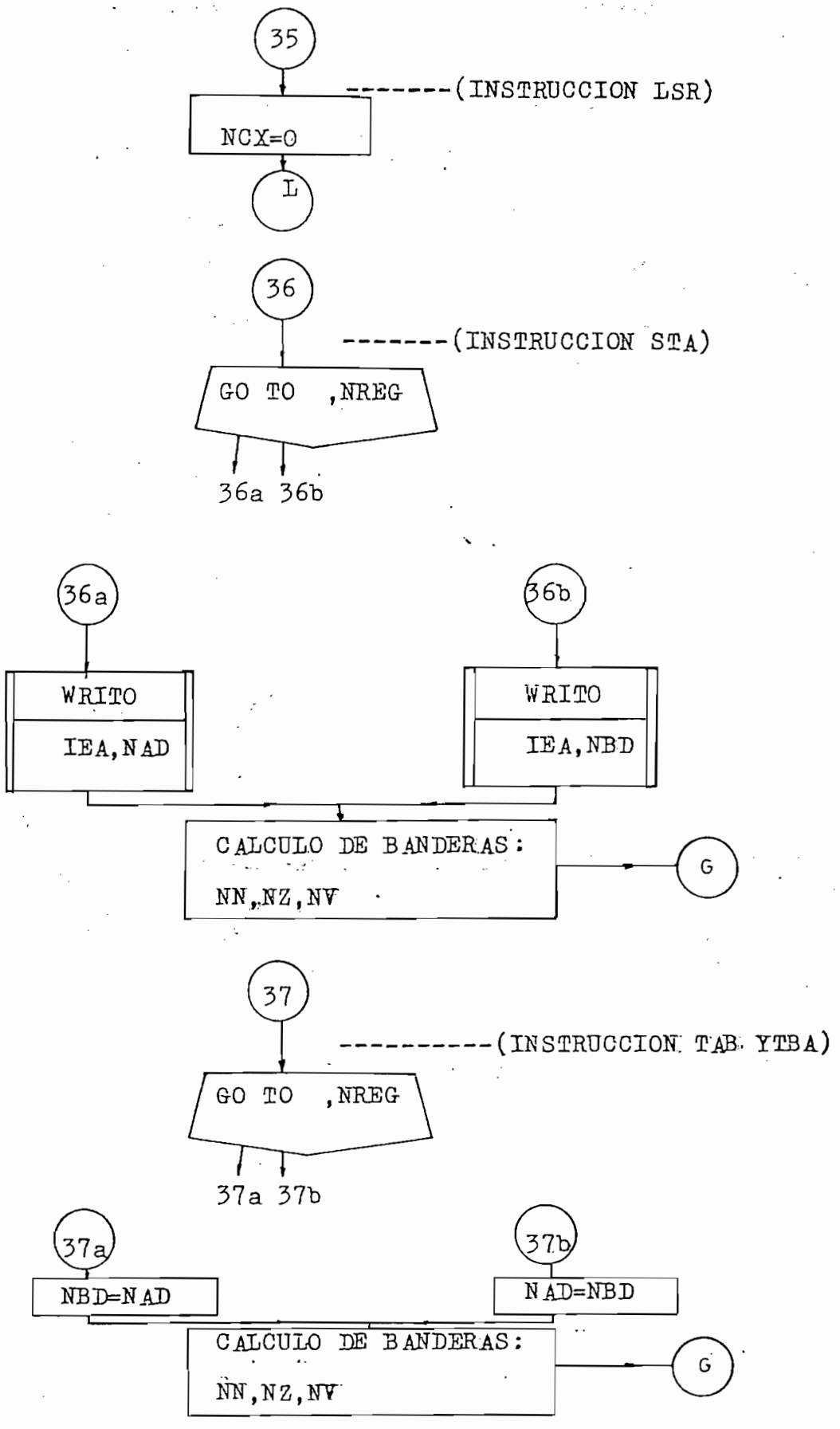






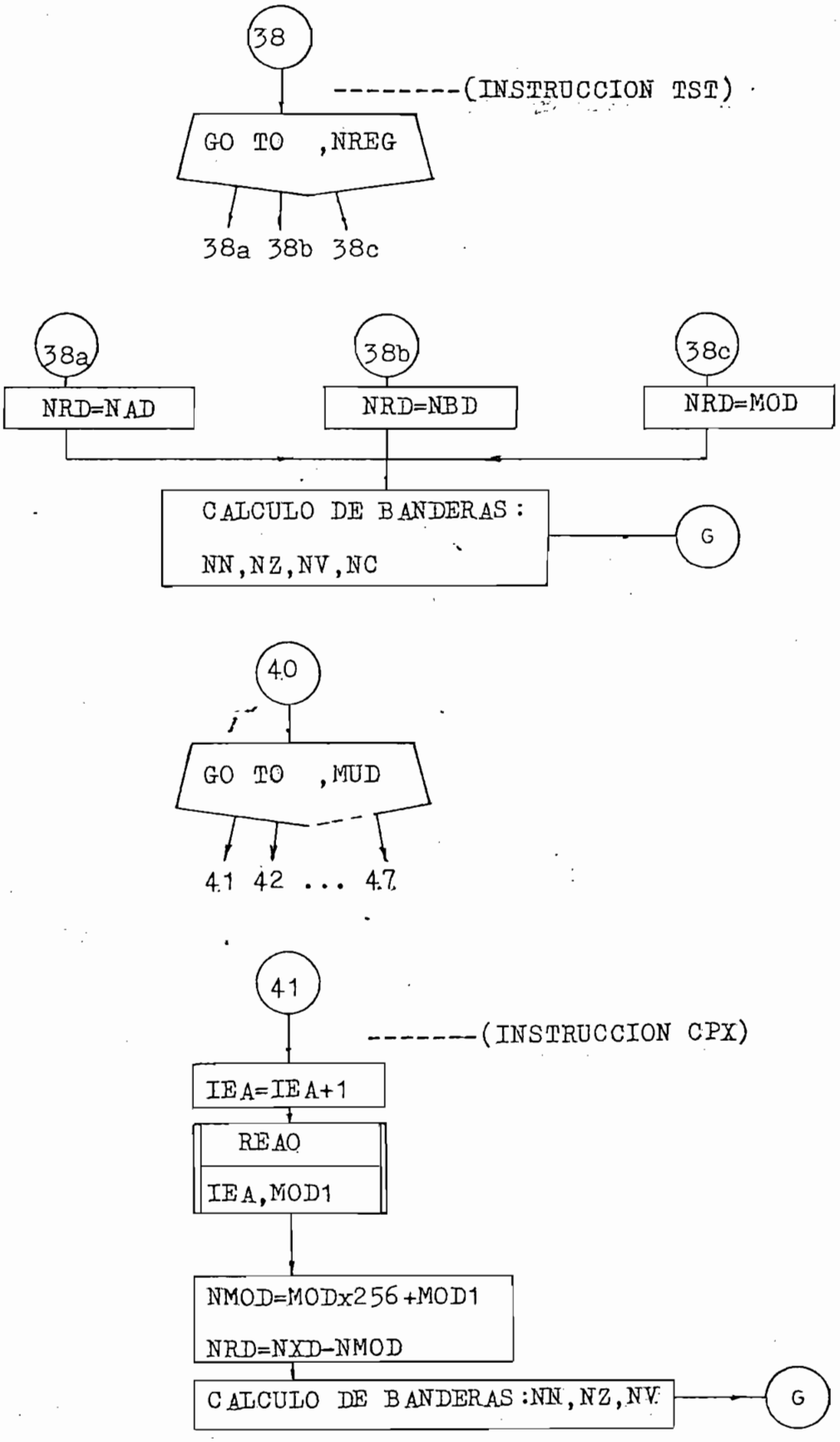
SUBRUTINA EJET

Continuación



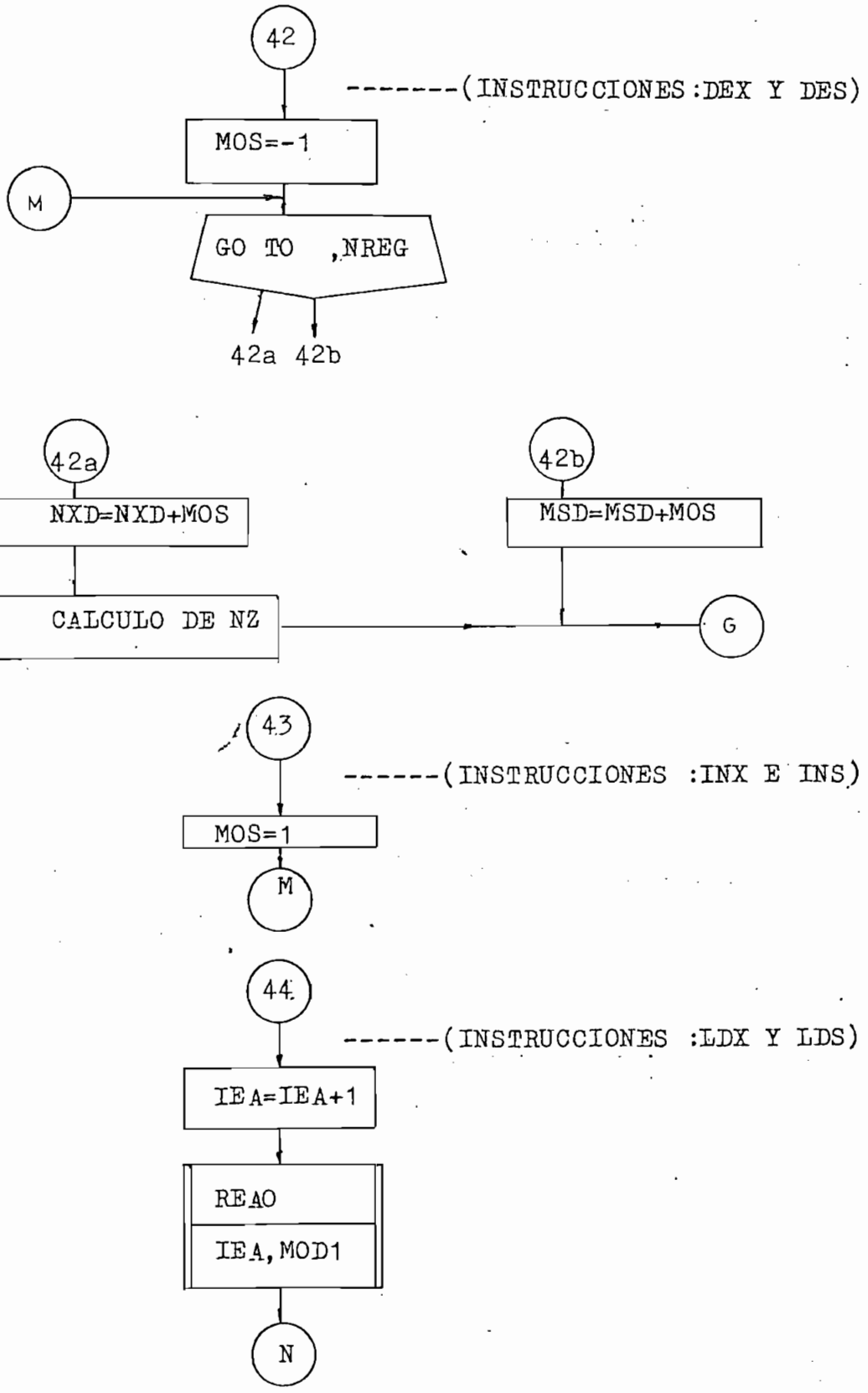
SUBRUTINA EJET

Continuación



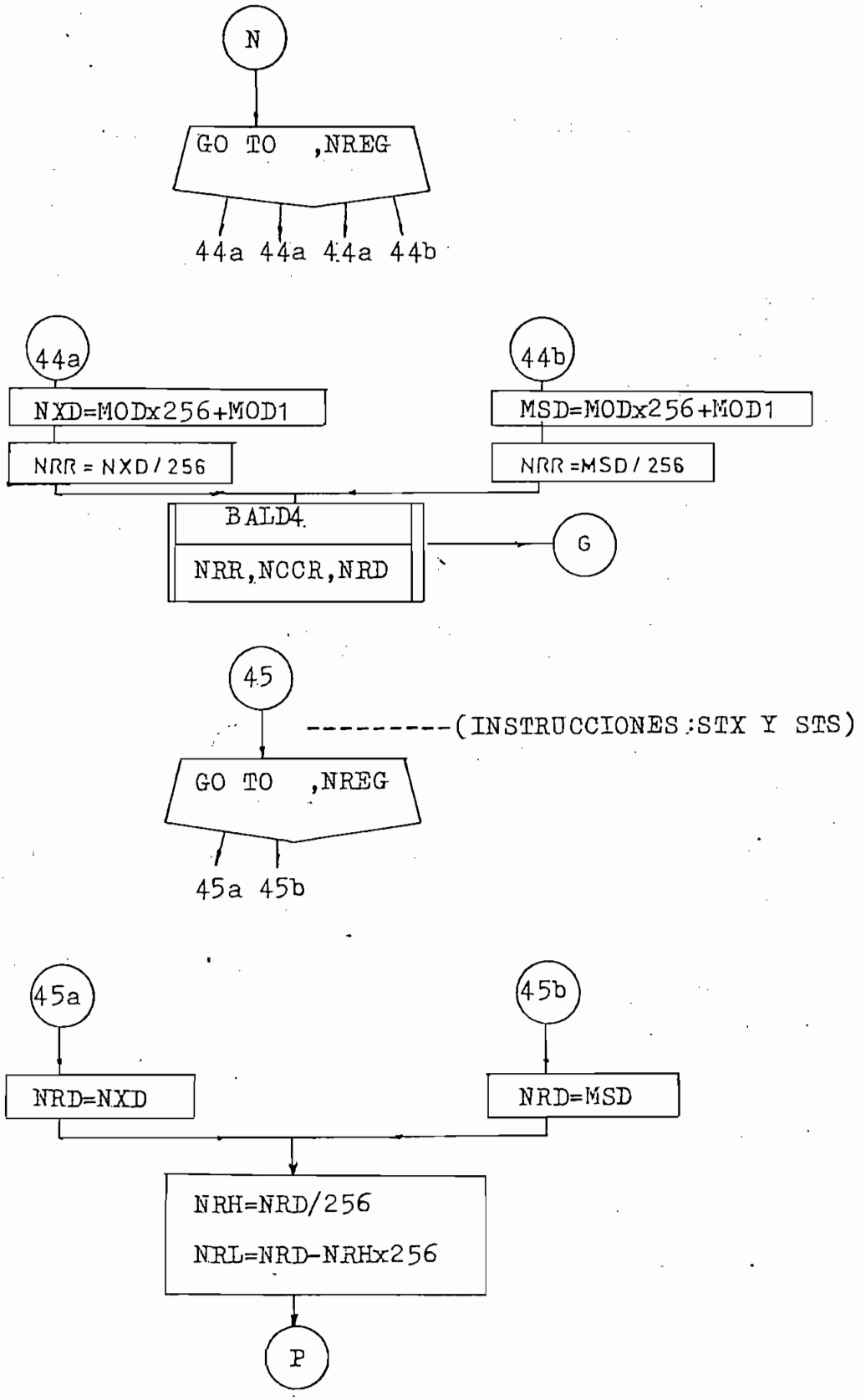
SUBRUTINA EJET

Continuación



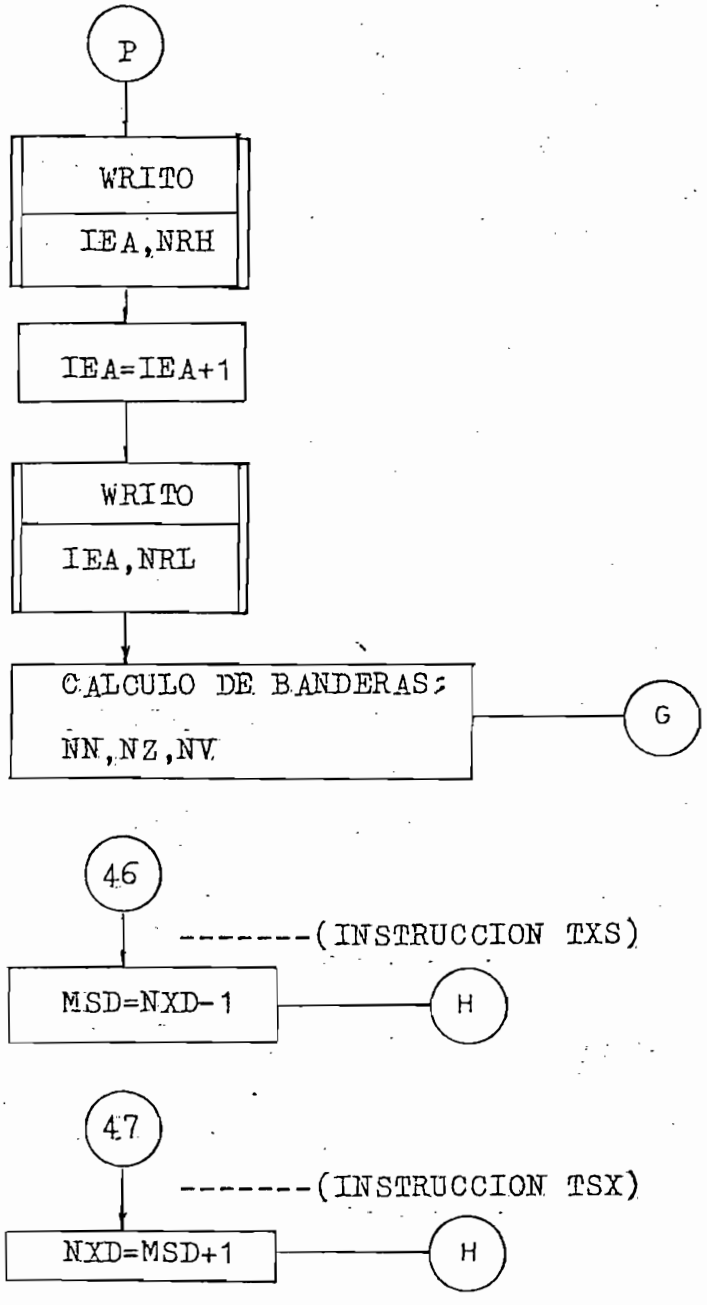
SUBRUTINA EJET

Continuación



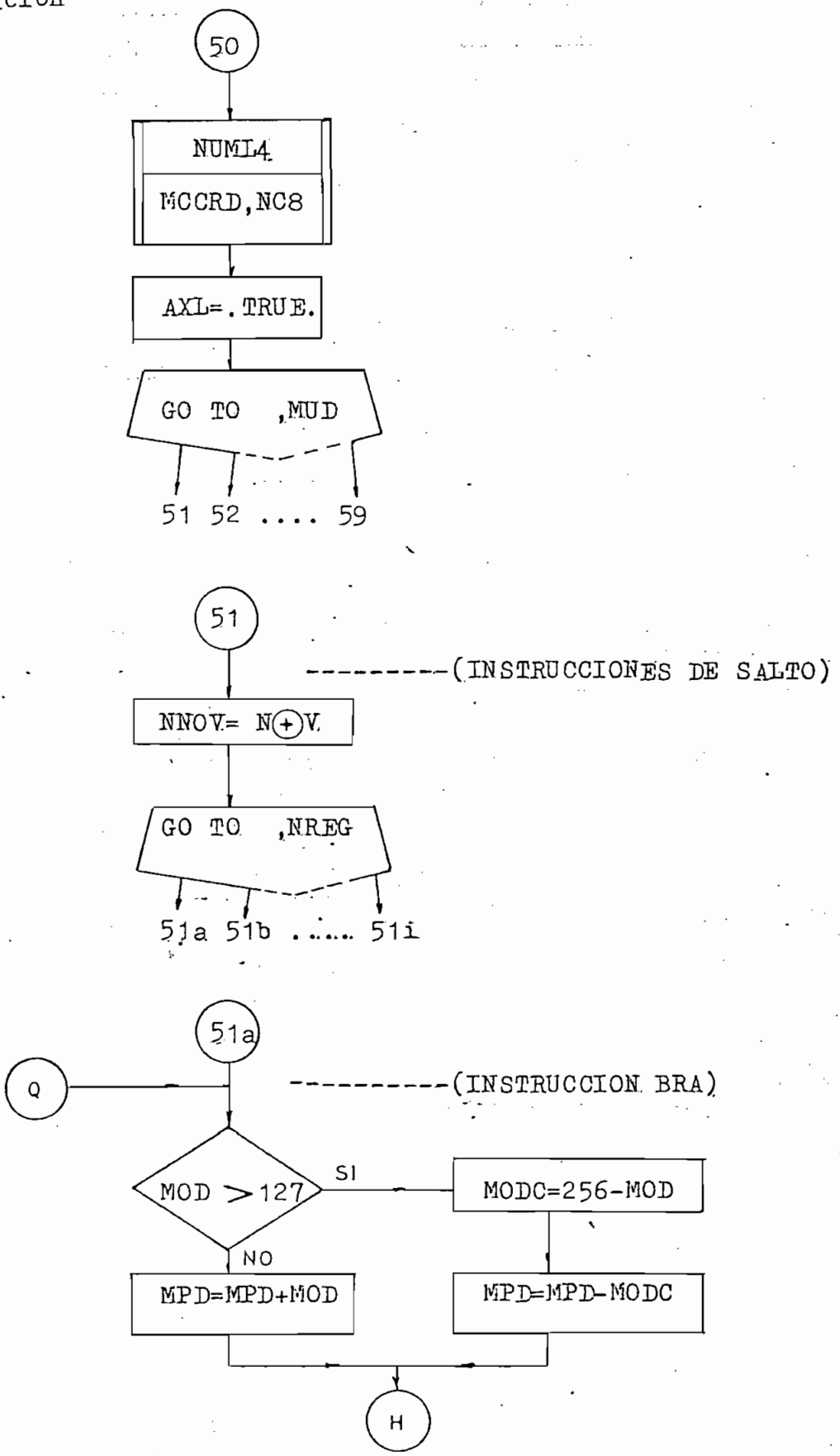
SUBRUTINA EJET

Continuación



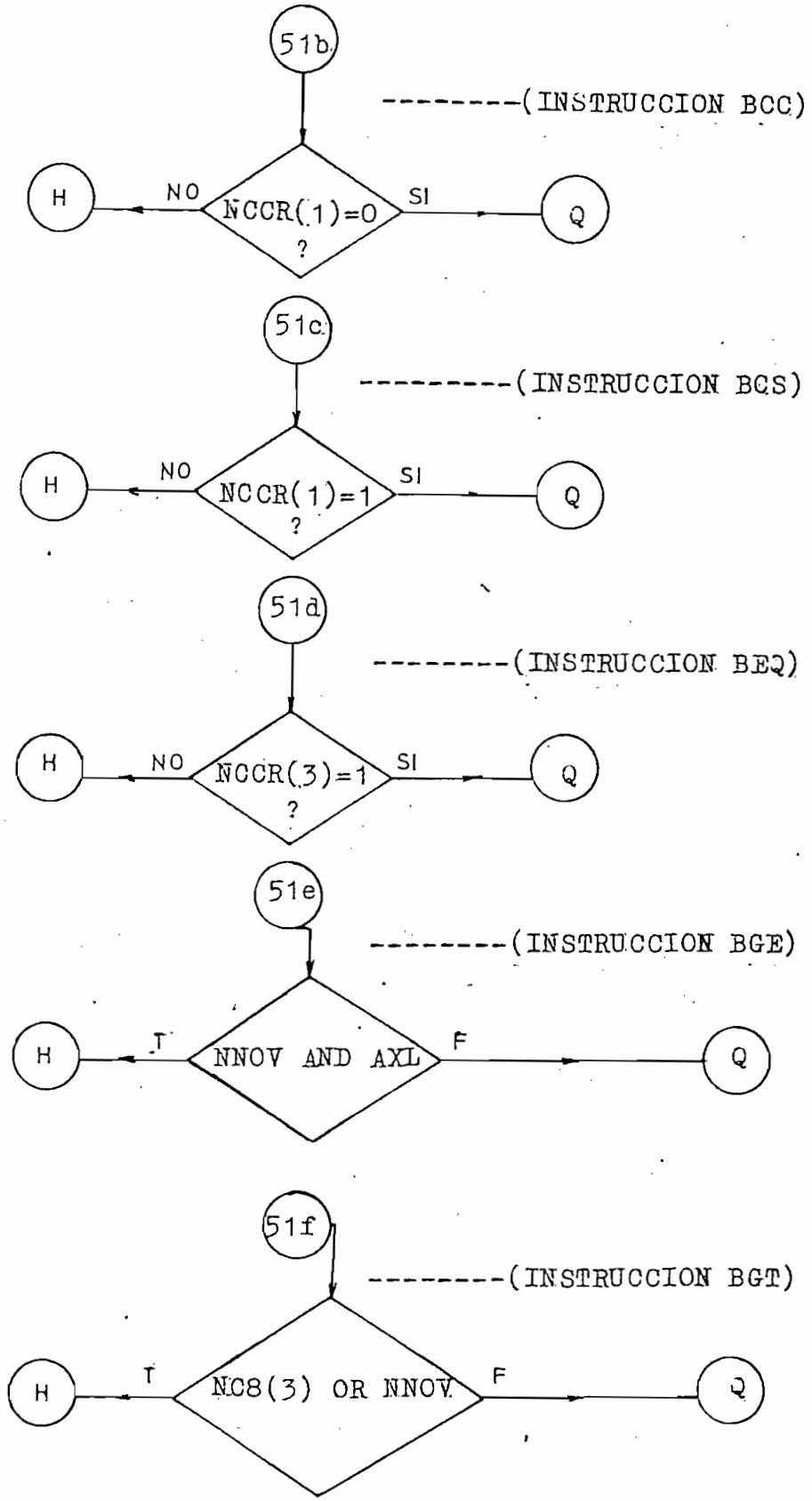
SUBRUTINA EJET

Continuación



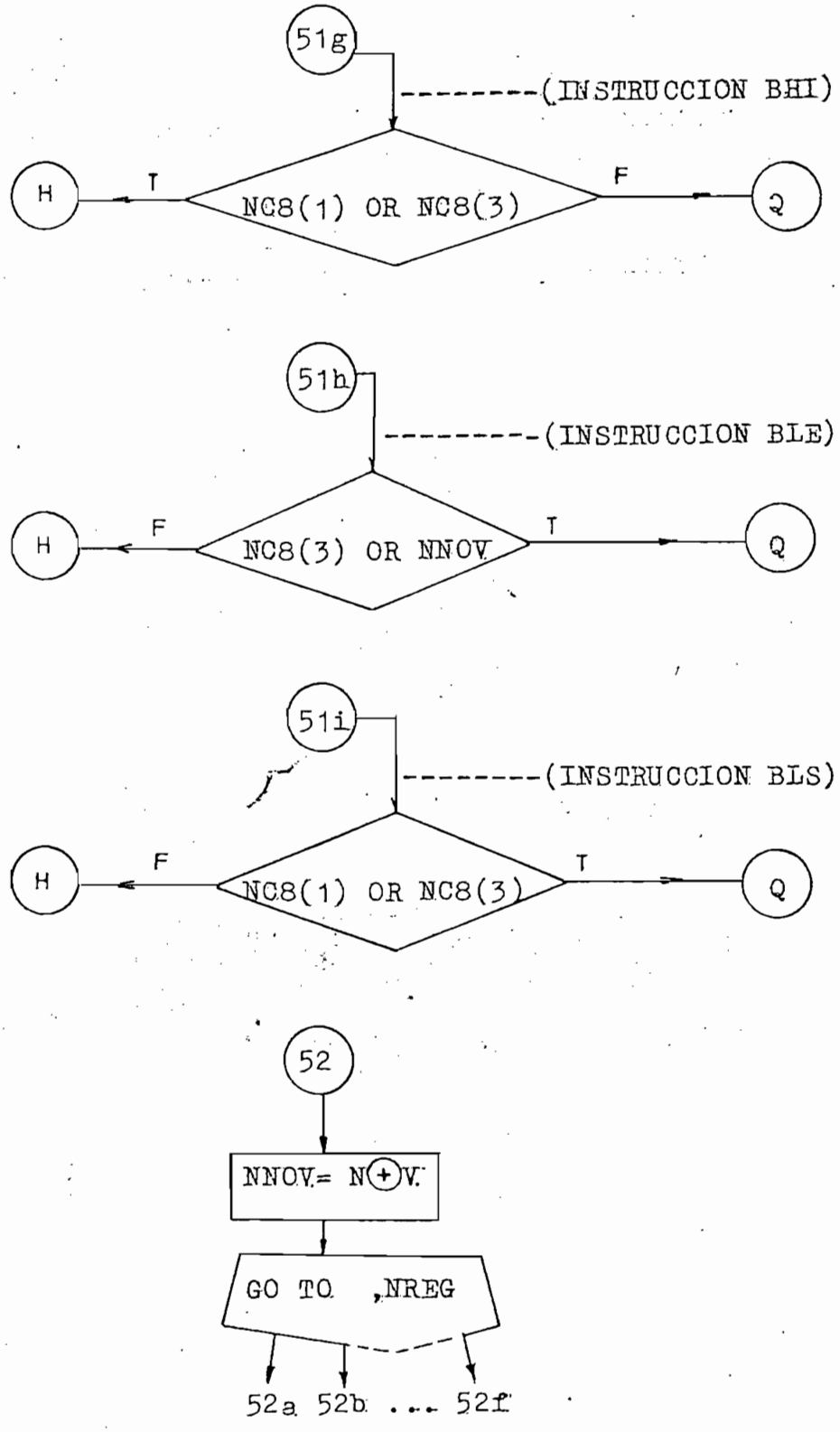
SUBRUTINA EJET

Continuación



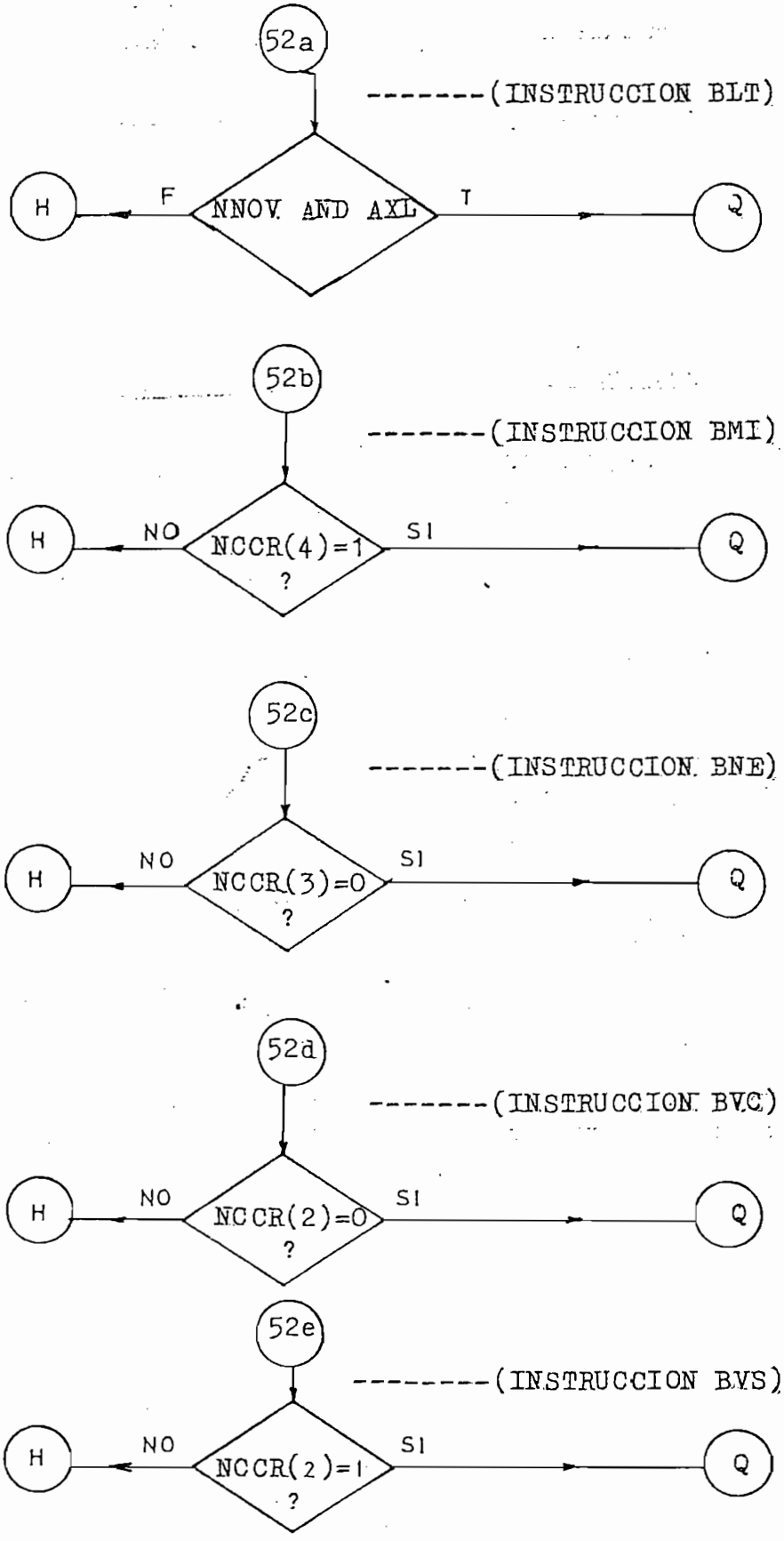
SUBRUTINA EJET

Continuación



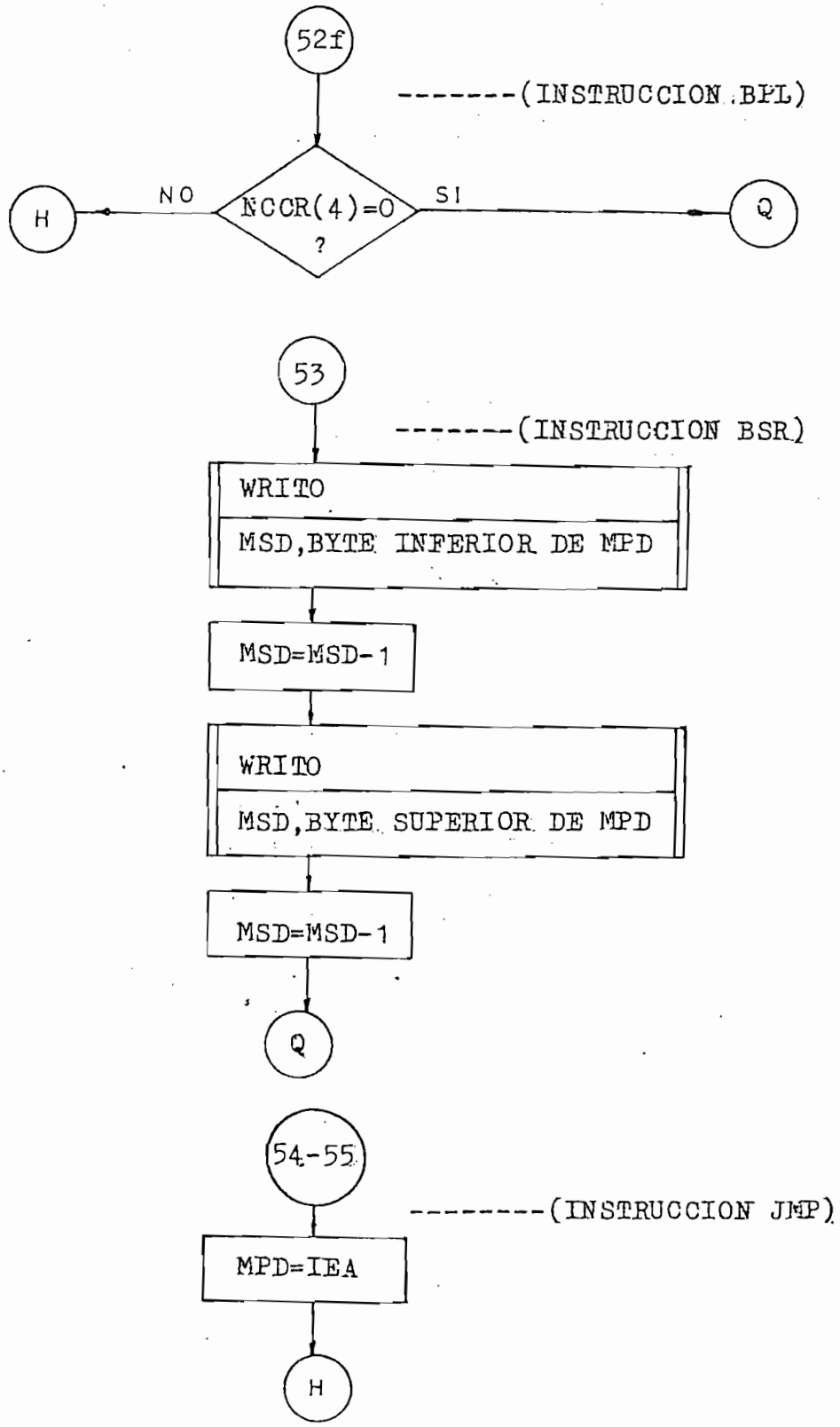
SUBRUTINA EJET

Continuación



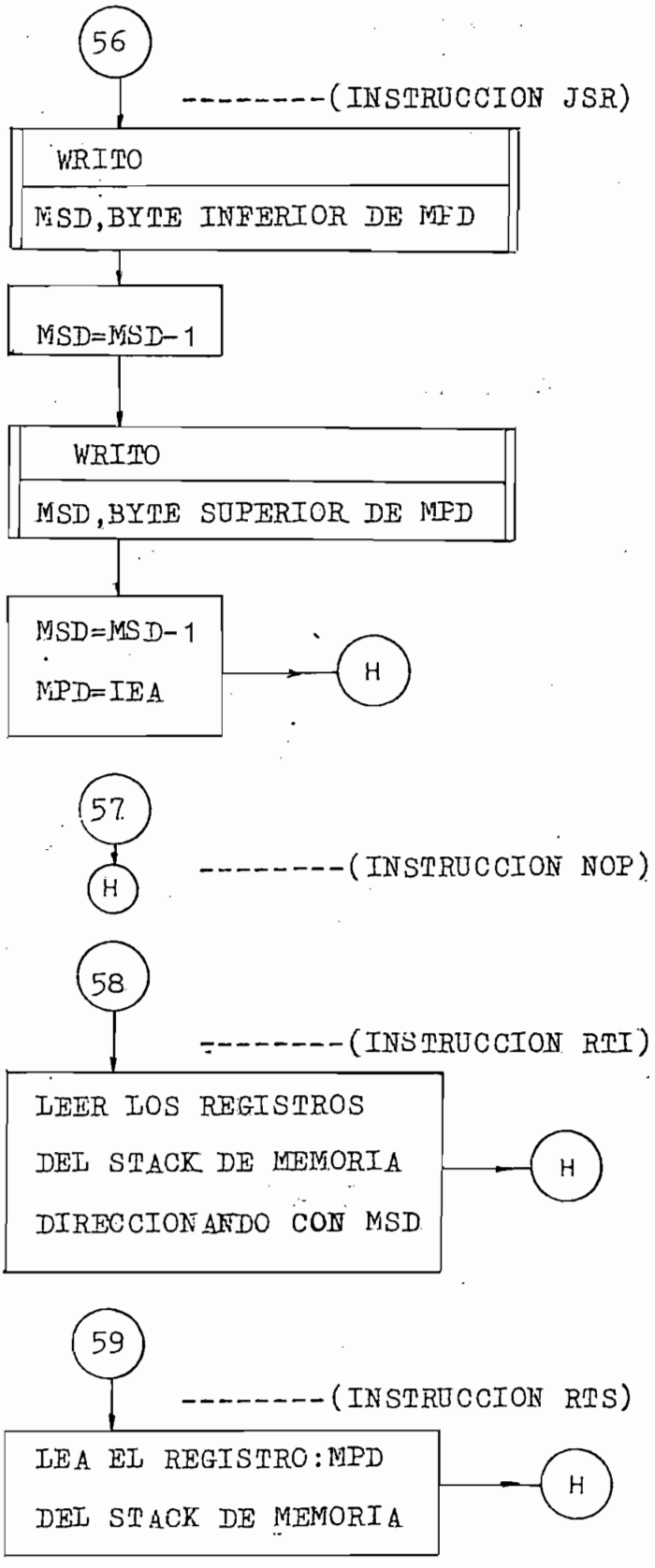
SUBRUTINA EJET

Continuación



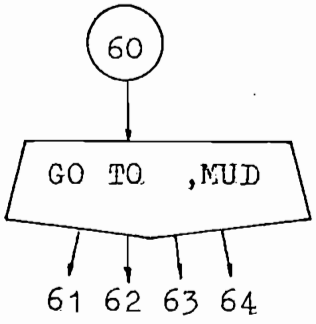
SUBRUTINA EJET

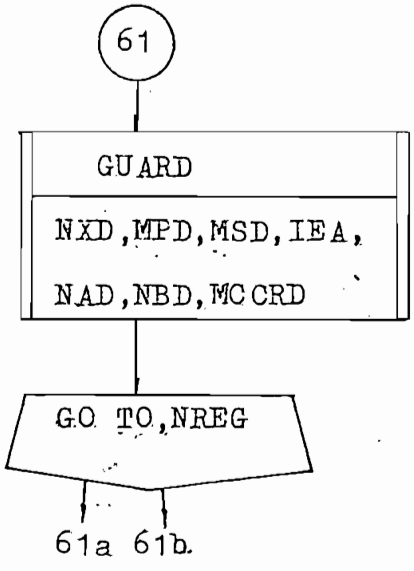
Continuación

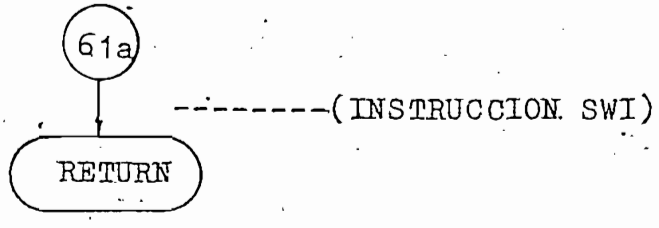


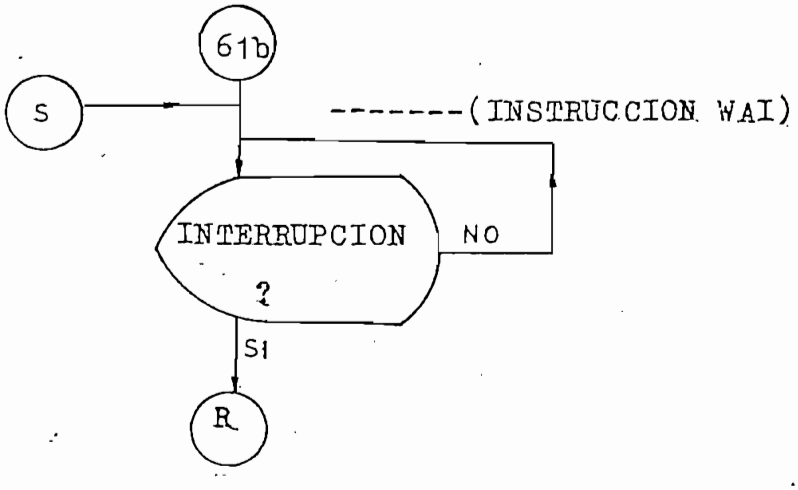
SUBRUTINA EJET

Continuación



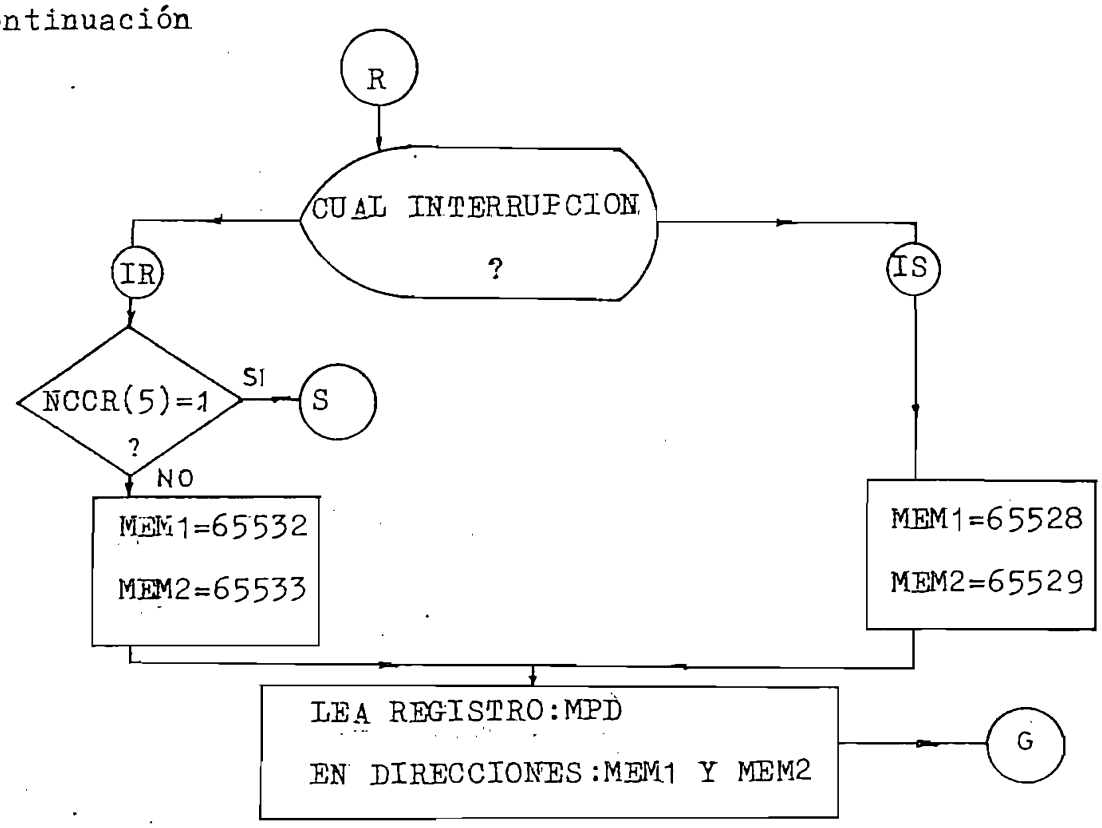


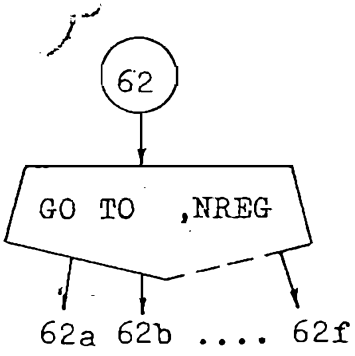


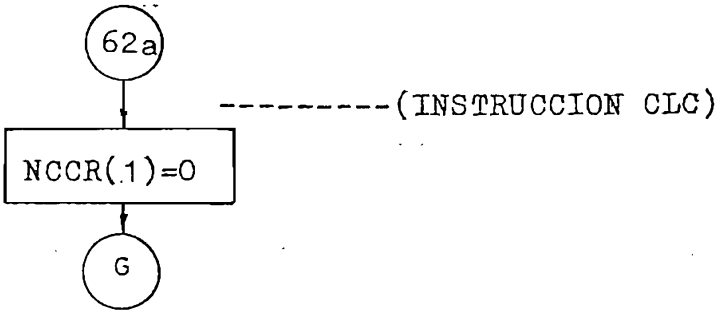


SUBRUTINA EJET

Continuación

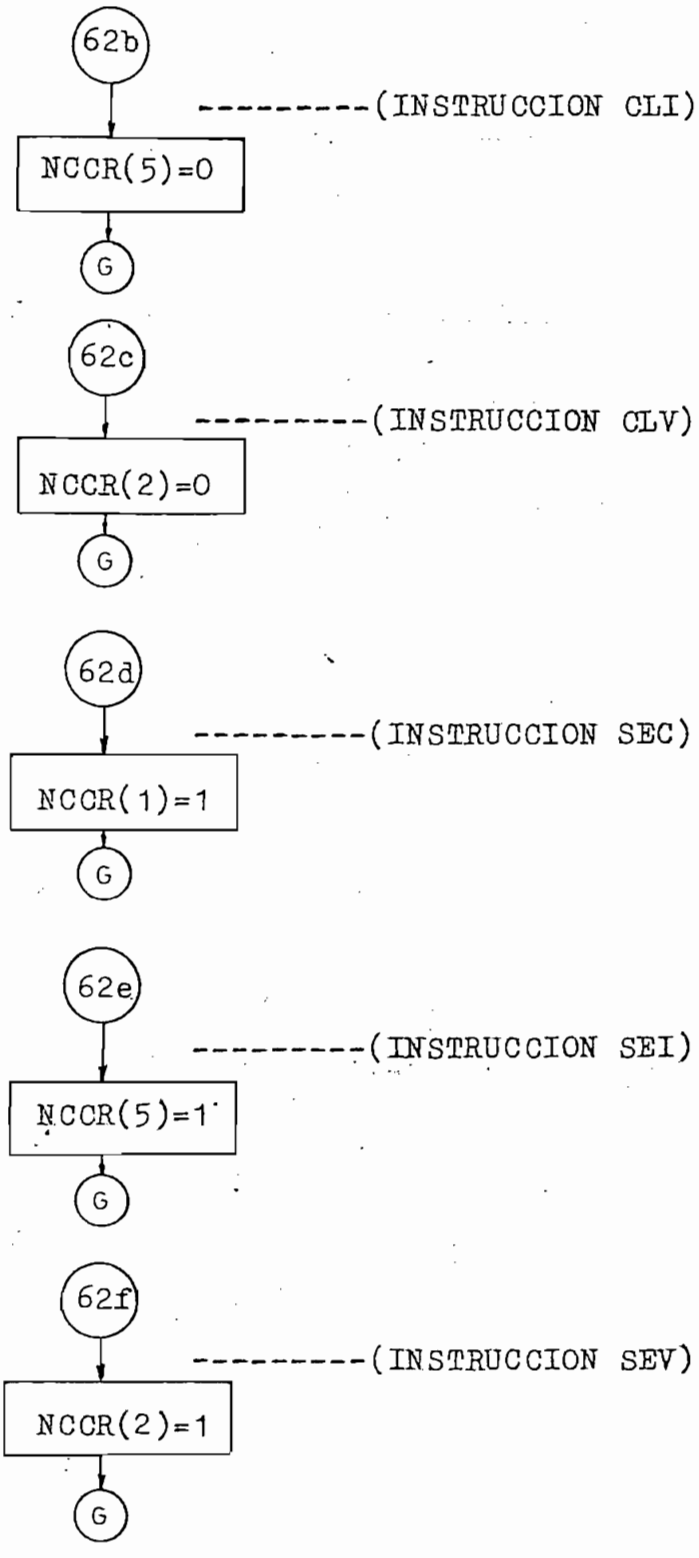






SUBRUTINA EJET

Continuación



SUBRUTINA EJET

Continuación

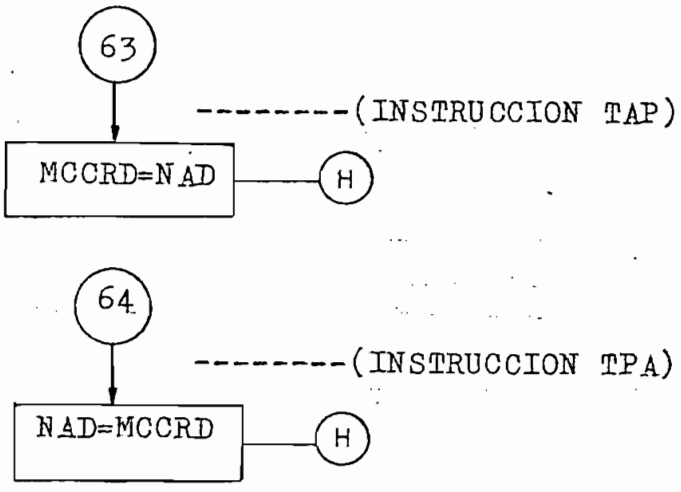


FIG. 4-4-27

SUBRUTINA WRITO

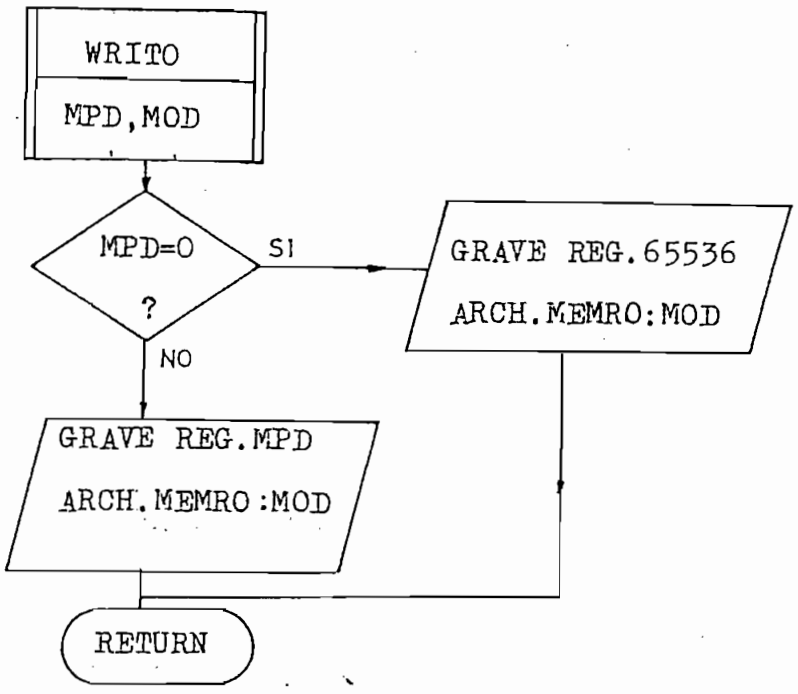


FIG. 4-4-28

SUBRUTINA ENRE

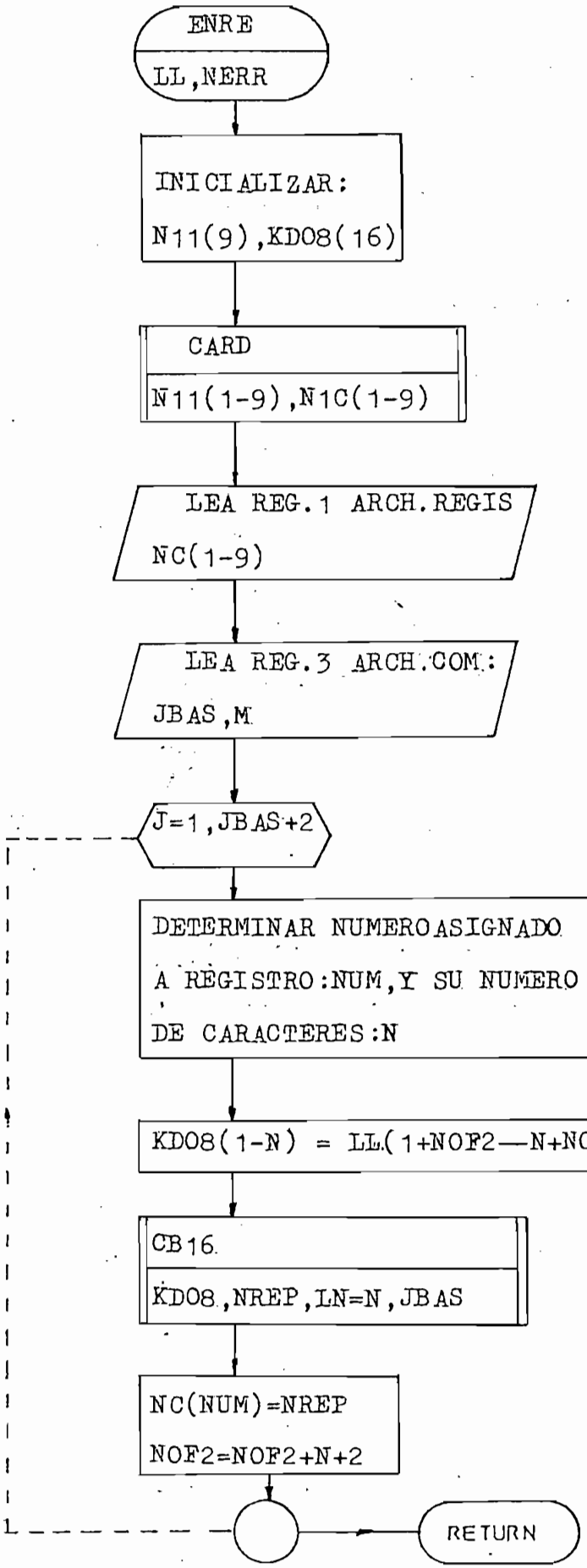


FIG. 4-4-29

SUBRUTINA ULTID

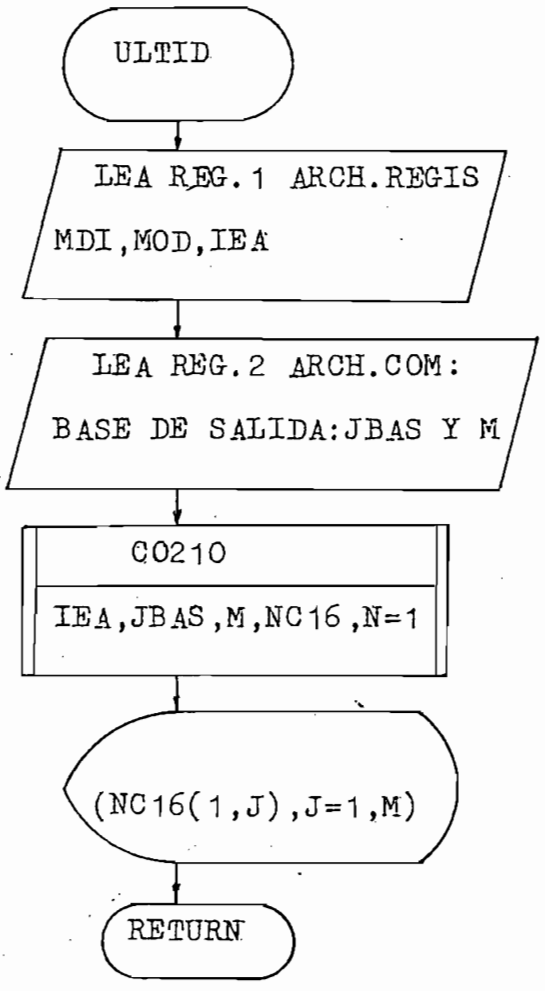


FIG. 4-4-30

SUBRUTINA EXPLO

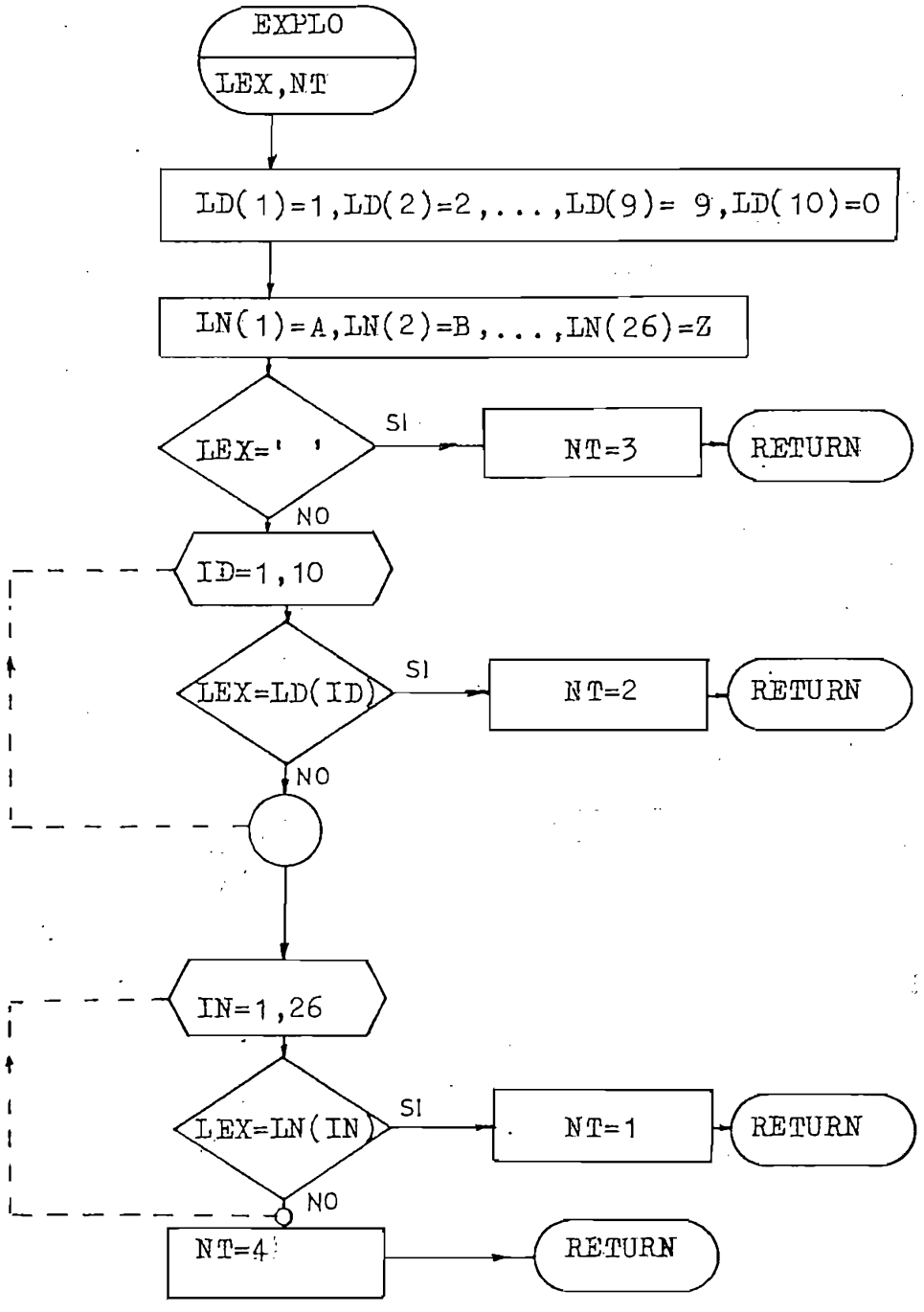


FIG. 4-4-31

SUBRUTINA CARD

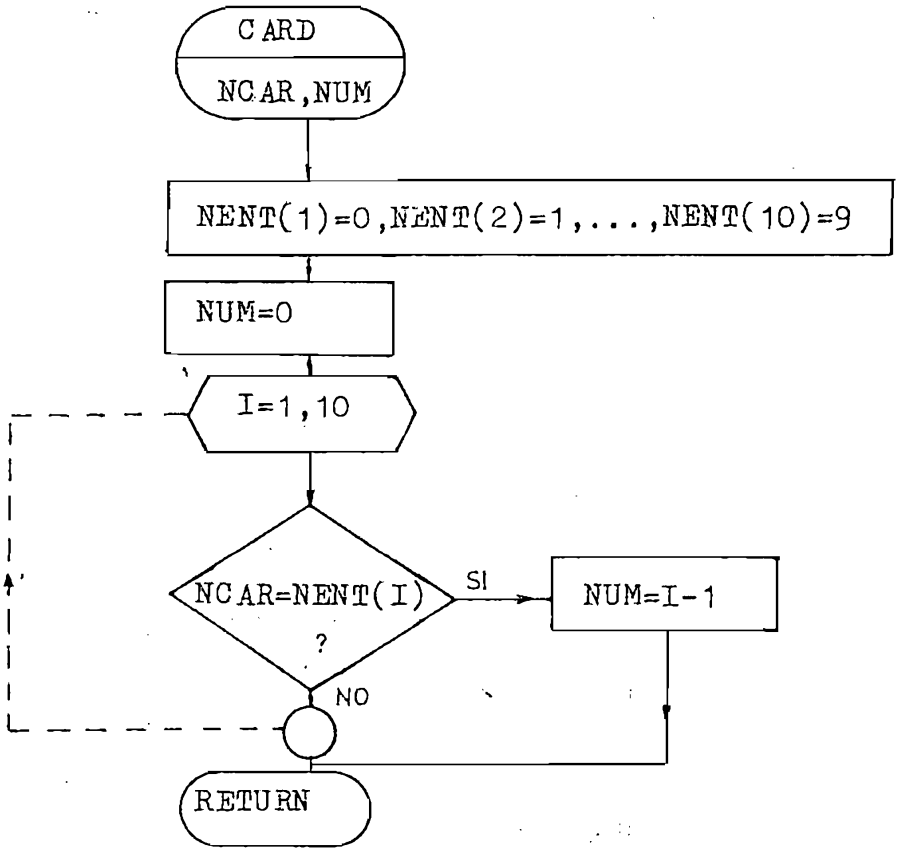


FIG.4-4-32

SUBRUTINA C0210

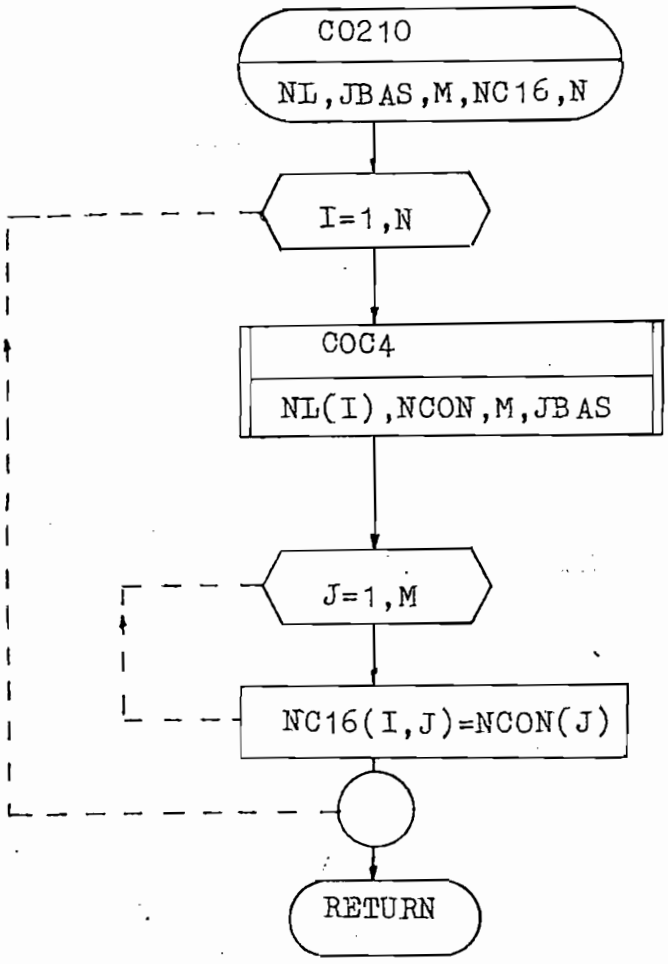


FIG.4-4-33

SUBRUTINA CB16

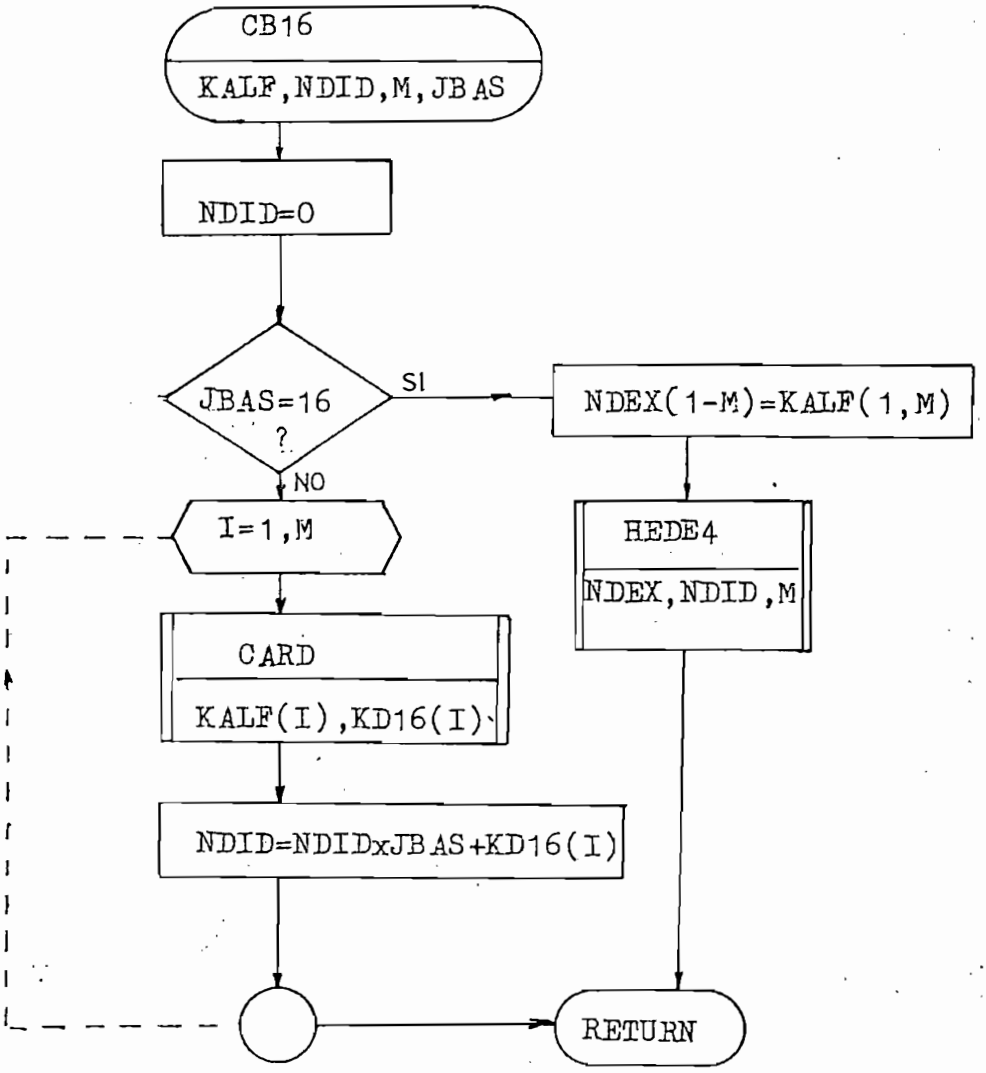


FIG. 4-4-34

SUBRUTINA COND4

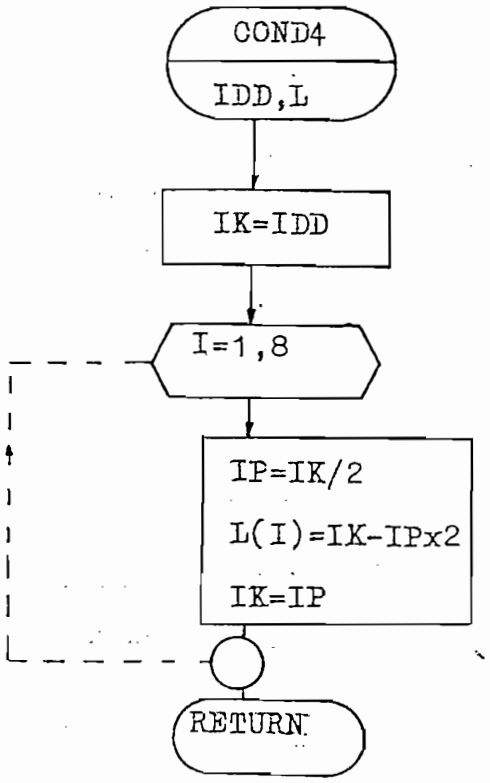


FIG. 4-4-35

SUBRUTINA NOMI

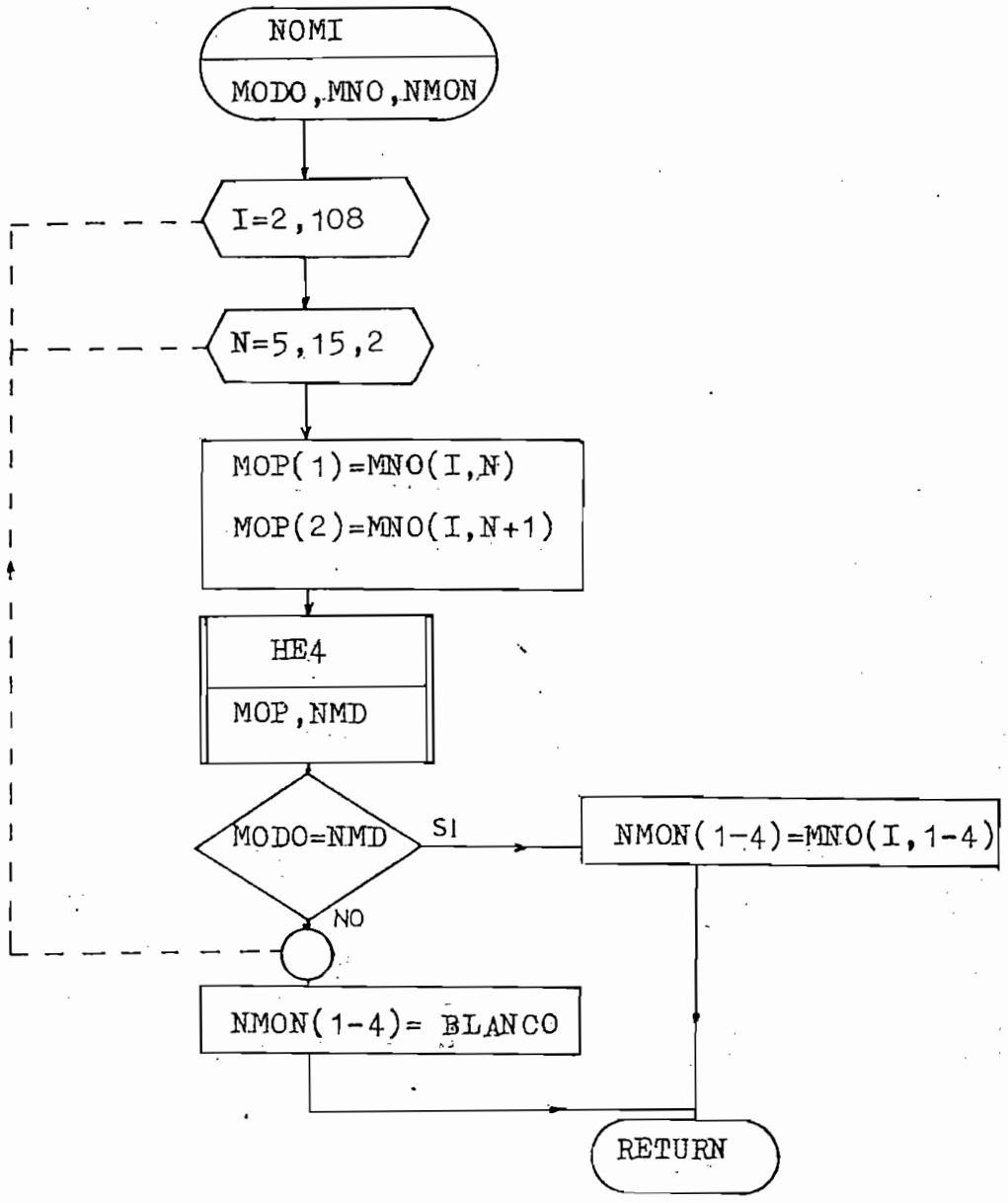


FIG.4-4-36

SUBRUTINA REDOS

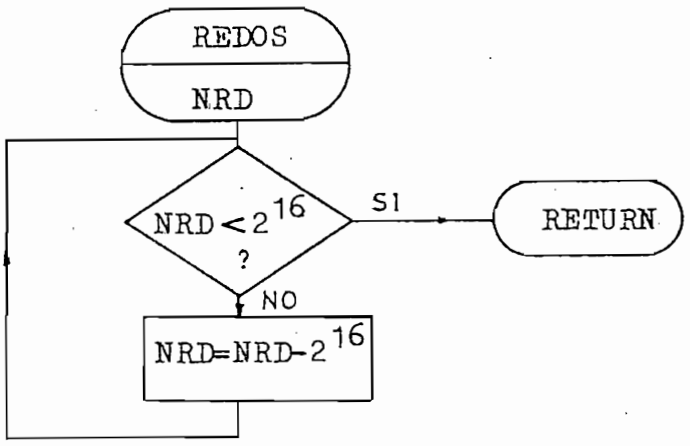


FIG. 4-4-37

SUBRUTINA NUML4

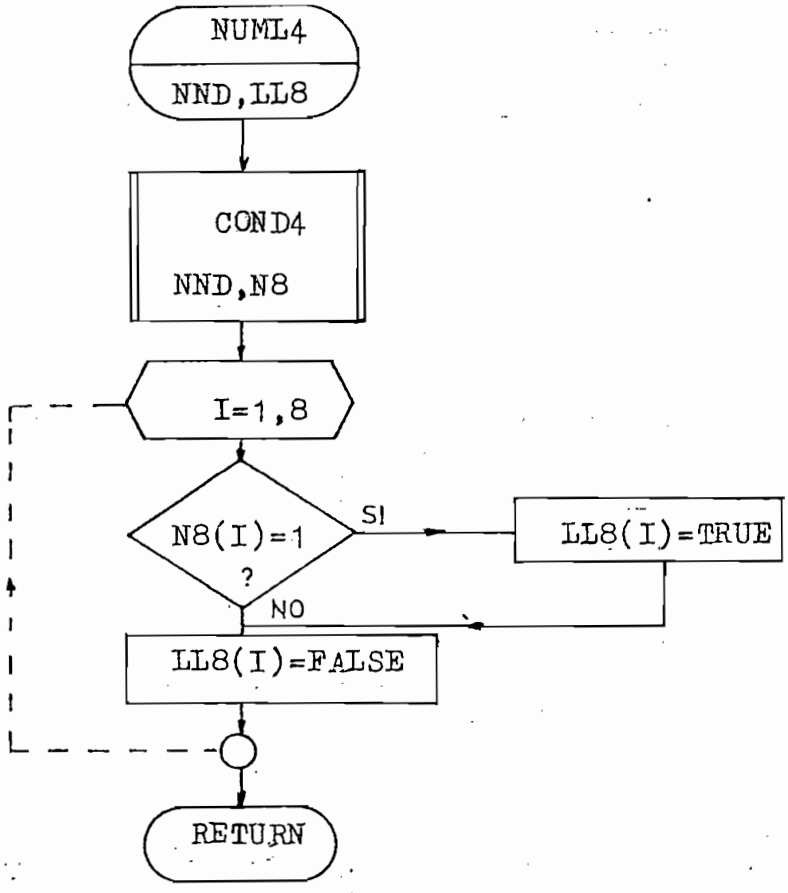


FIG.4-4-38

SUBRUTINA ME254

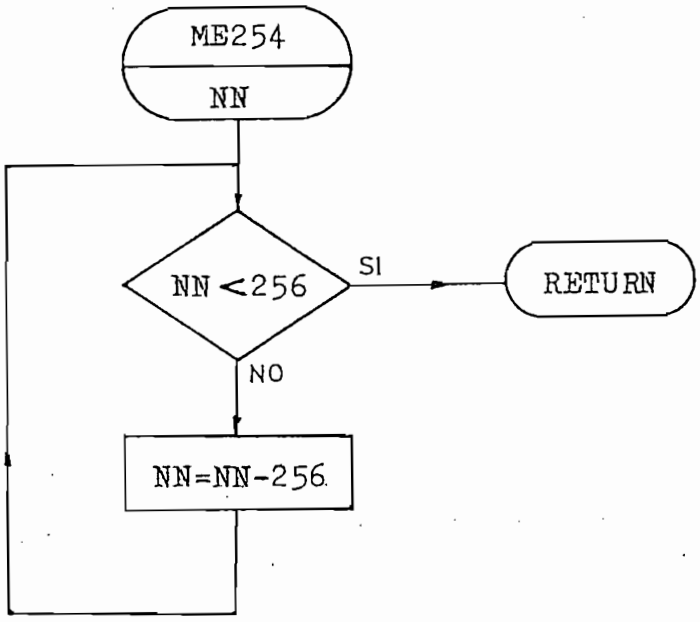


FIG. 4-4-39

SUBRUTINA HAC4

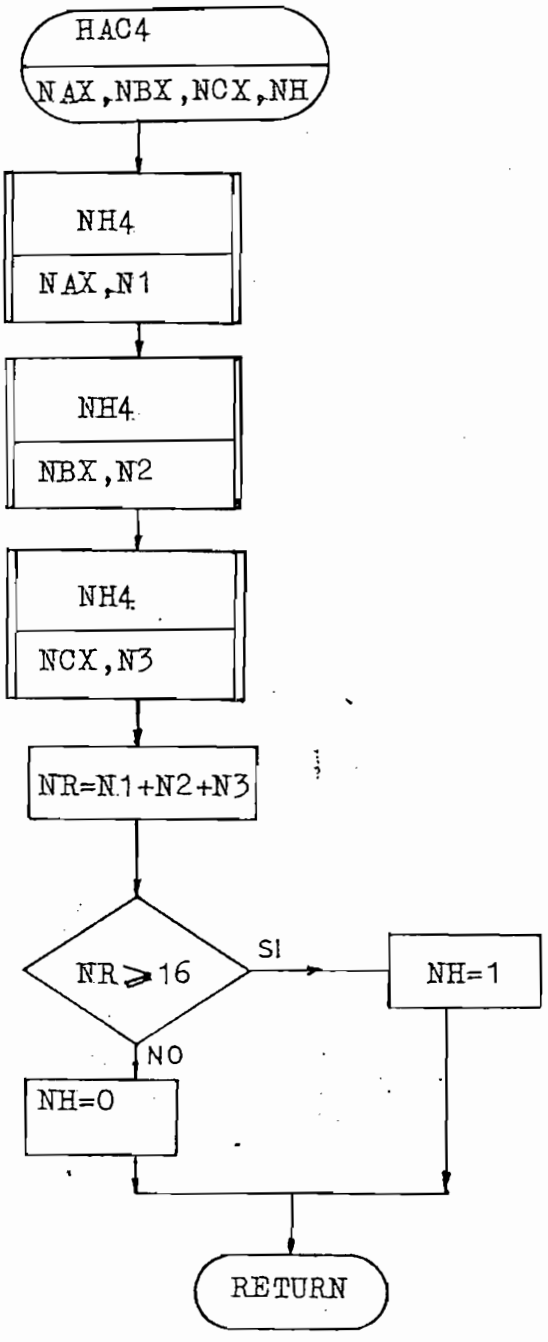


FIG.4-4-40

SUBRUTINA CR4

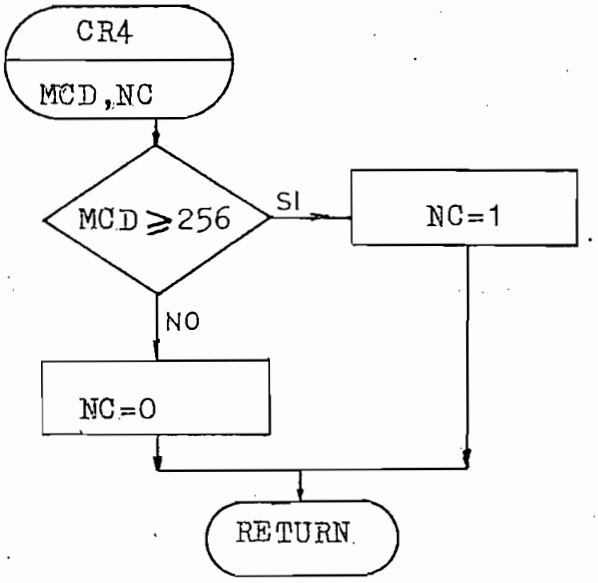


FIG. 4-4-41

SUBRUTINA BORR4

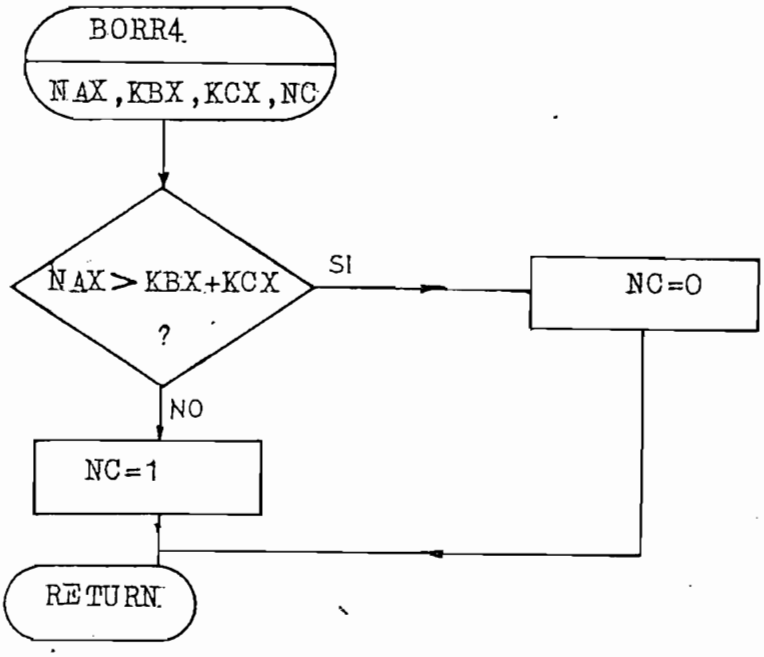


FIG. 4-4-42

SUBRUTINA NE4

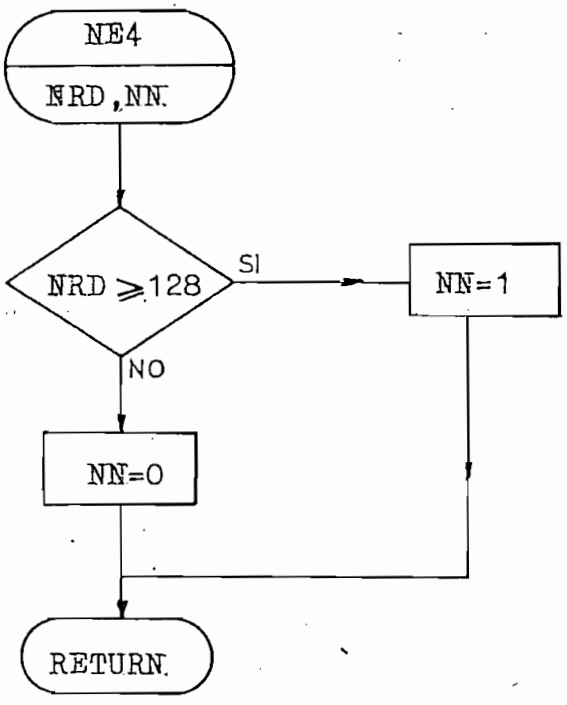


FIG.4-4-43

SUBRUTINA CER4

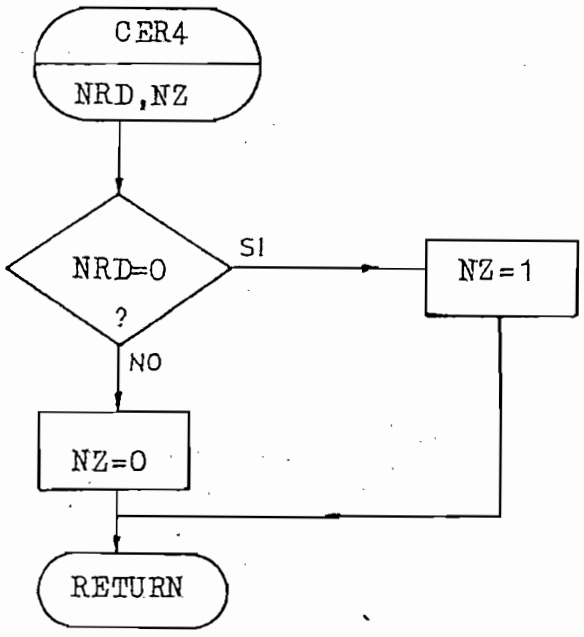


FIG.4-4-44

SUBRUTINA SOBR4

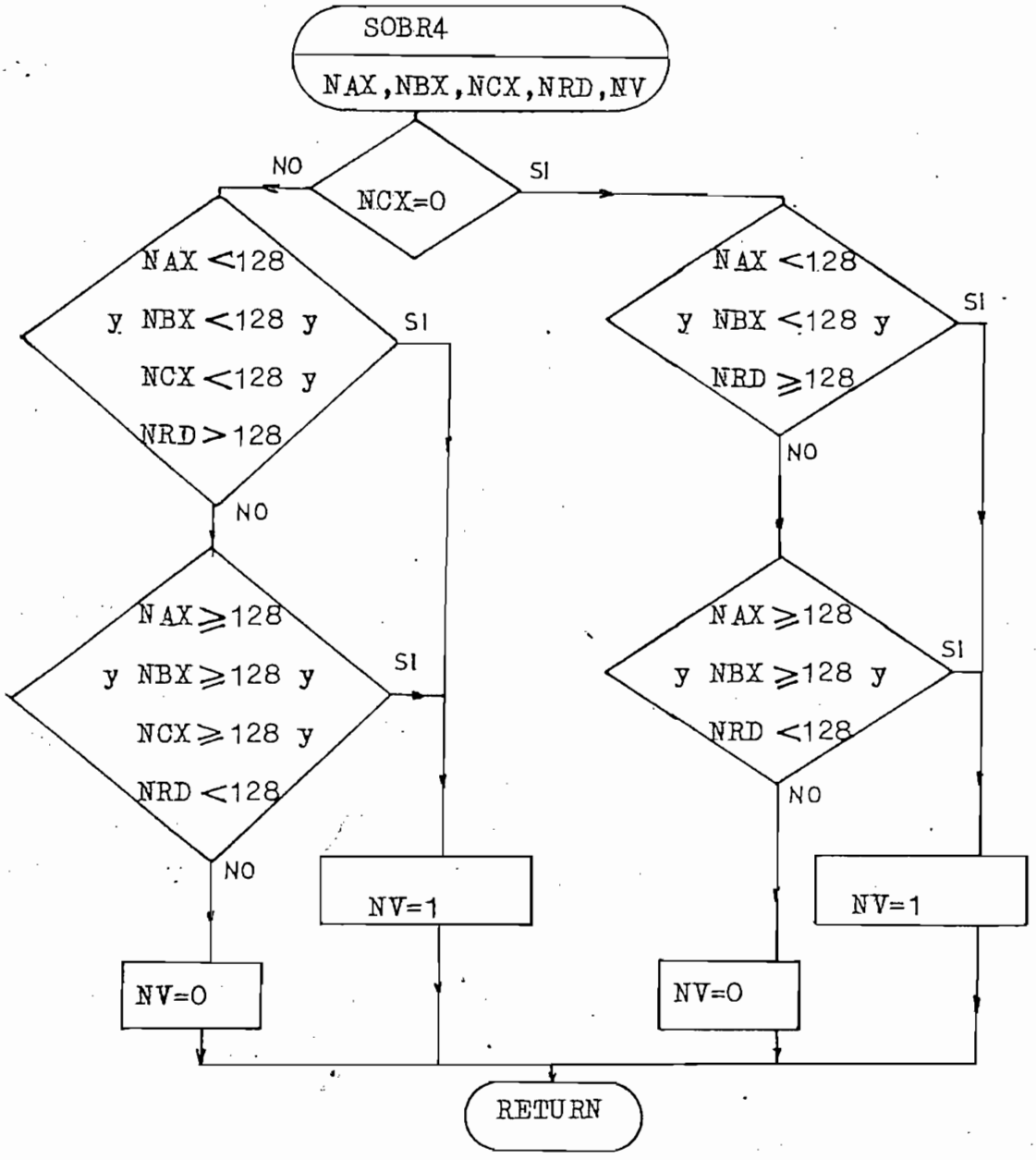


FIG.4-4-45

SUBRUTINA CONB4

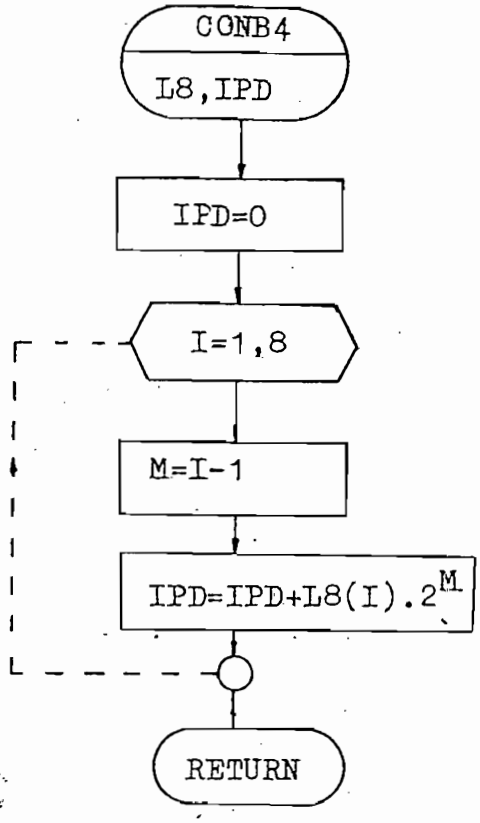


FIG.4-4-46

SUBRUTINA ARRE4

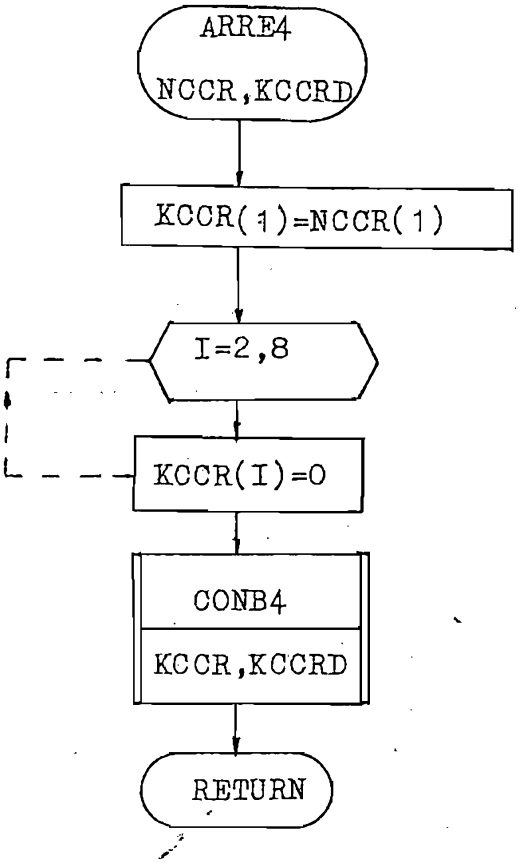


FIG.4-4-47

SUBRUTINA LOGN4

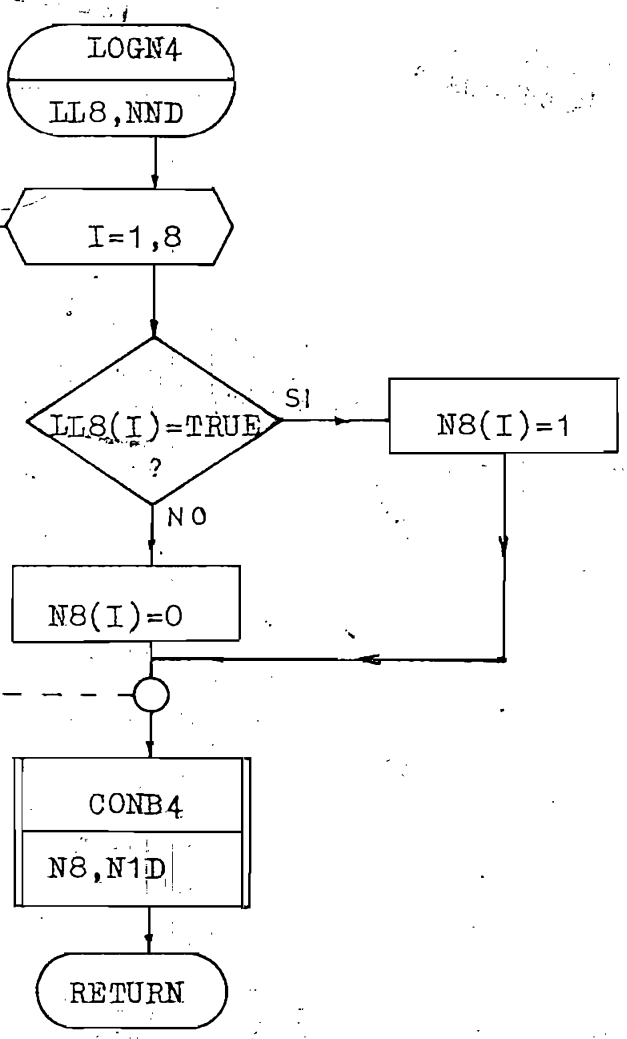


FIG.4-4-48

SUBRUTINA SOB4

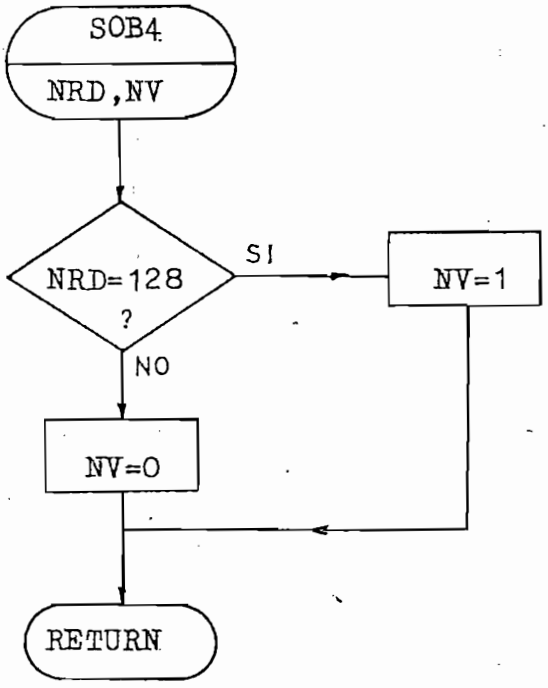


FIG. 4-4-49

SUBRUTINA CAR4

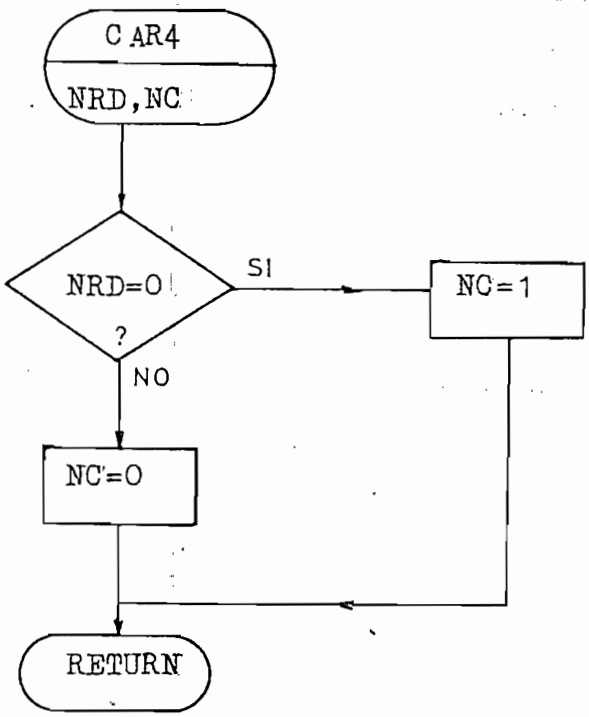


FIG. 4-4-50

SUBROUTINE R04

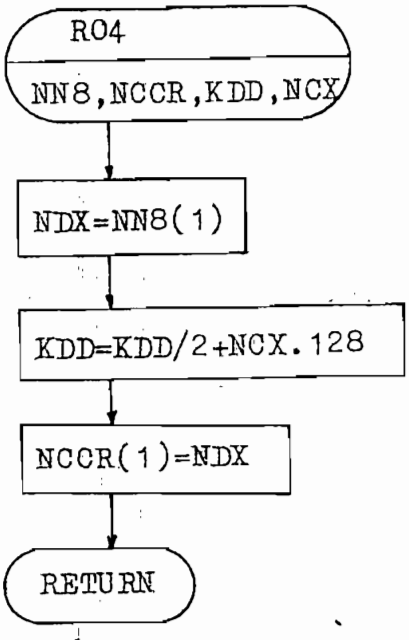


FIG.4-4-51

SUBRUTINA BALD4

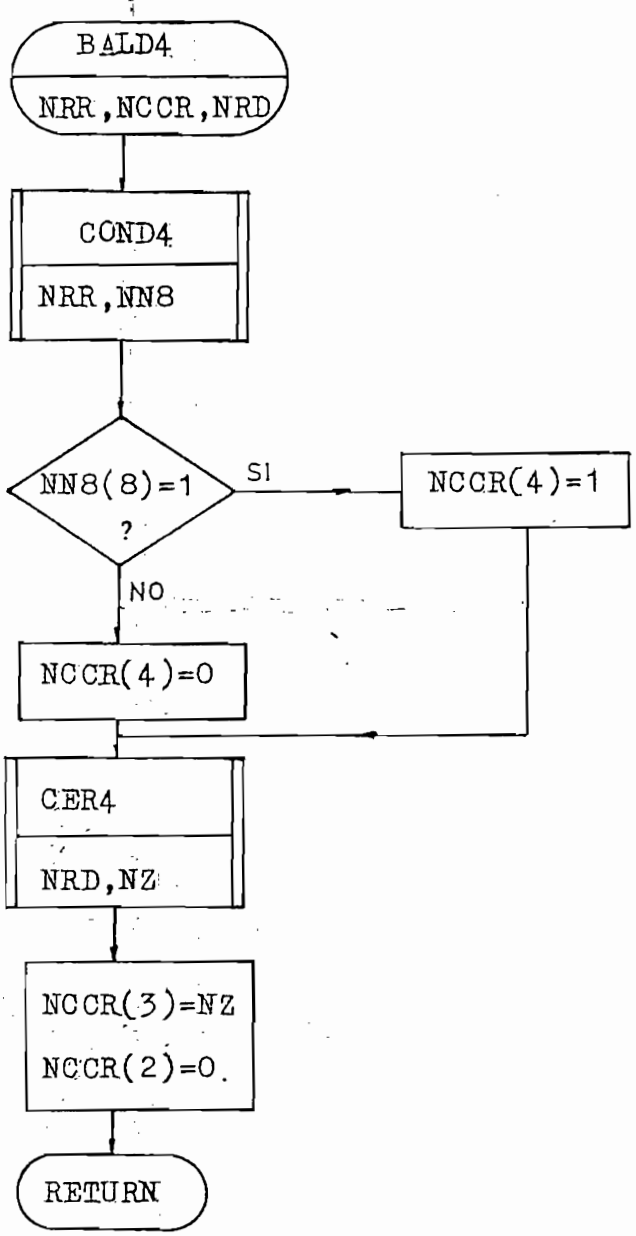


FIG. 4-4-52

SUBRUTINA REMOS

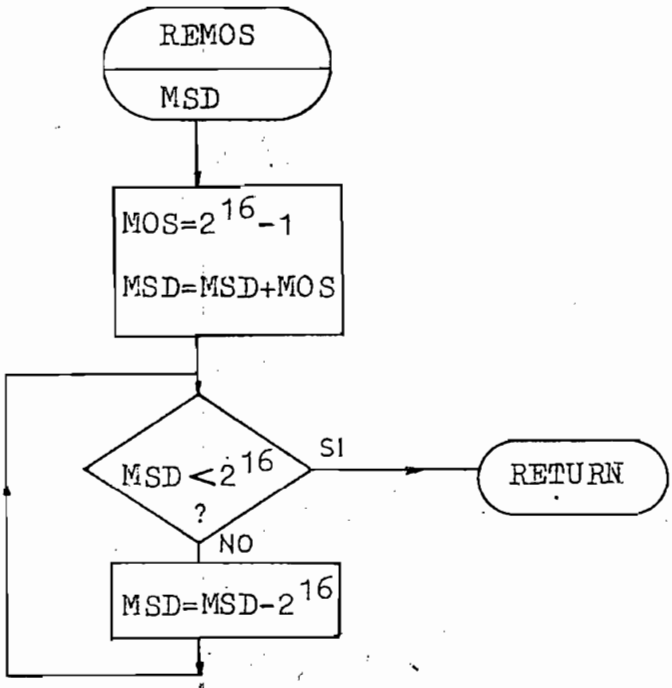


FIG.4-4-53

SUBRUTINA GUARD

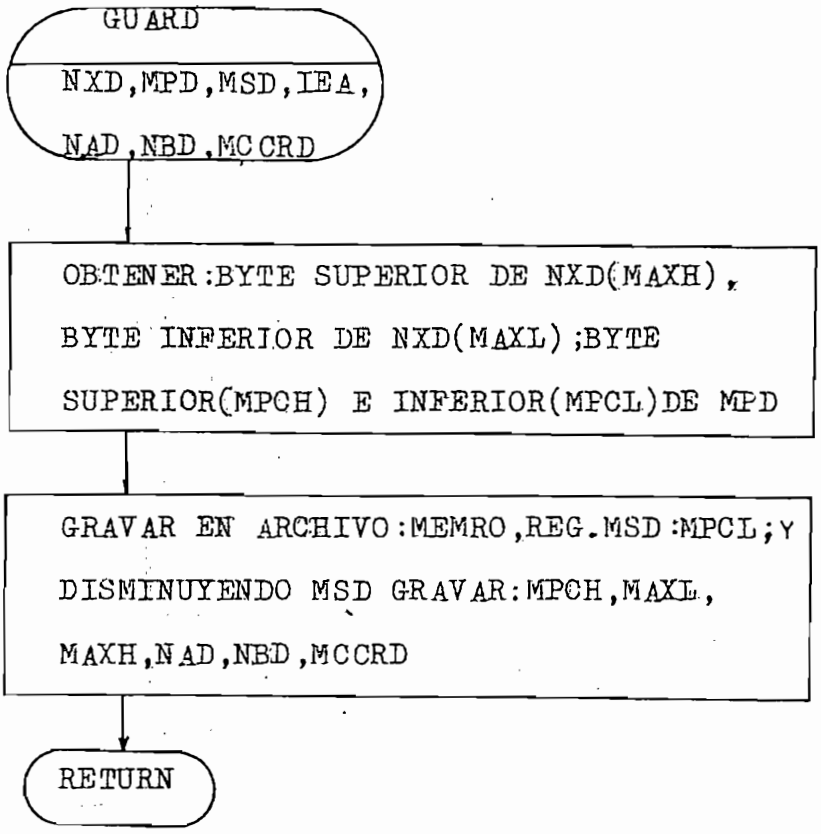


FIG.4-4-54

SUBRUTINA COC4

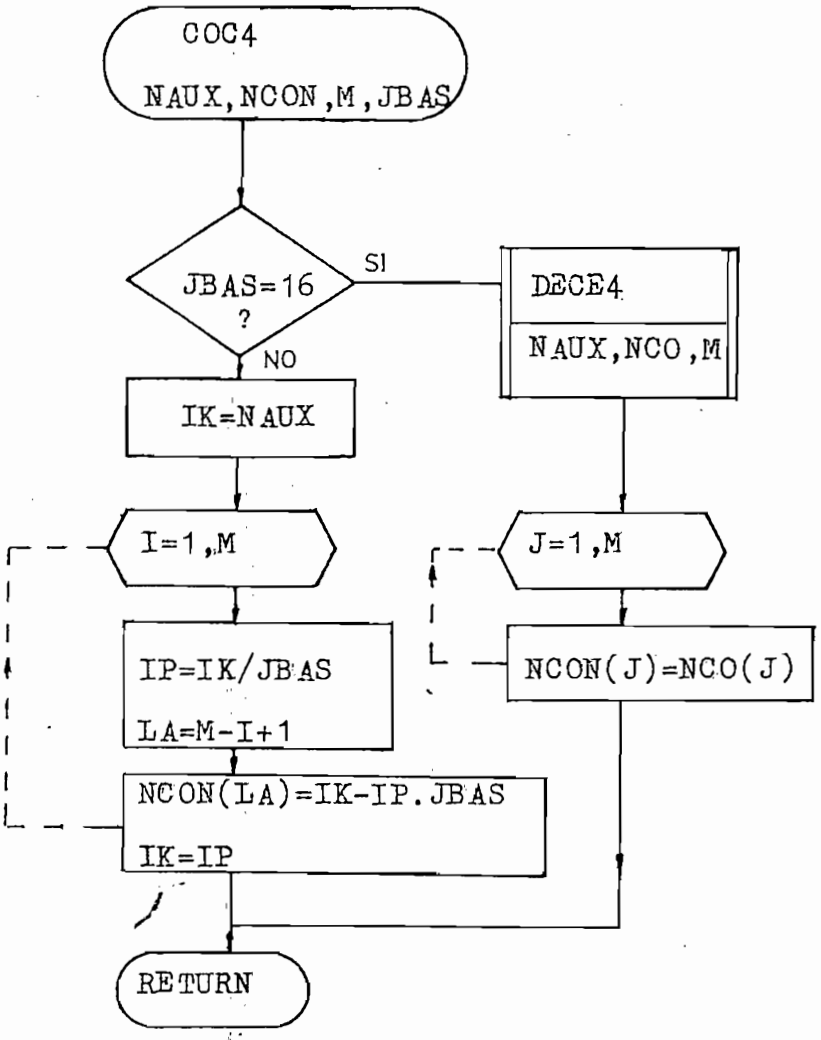


FIG.4-4-55

SUBRUTINA NH4

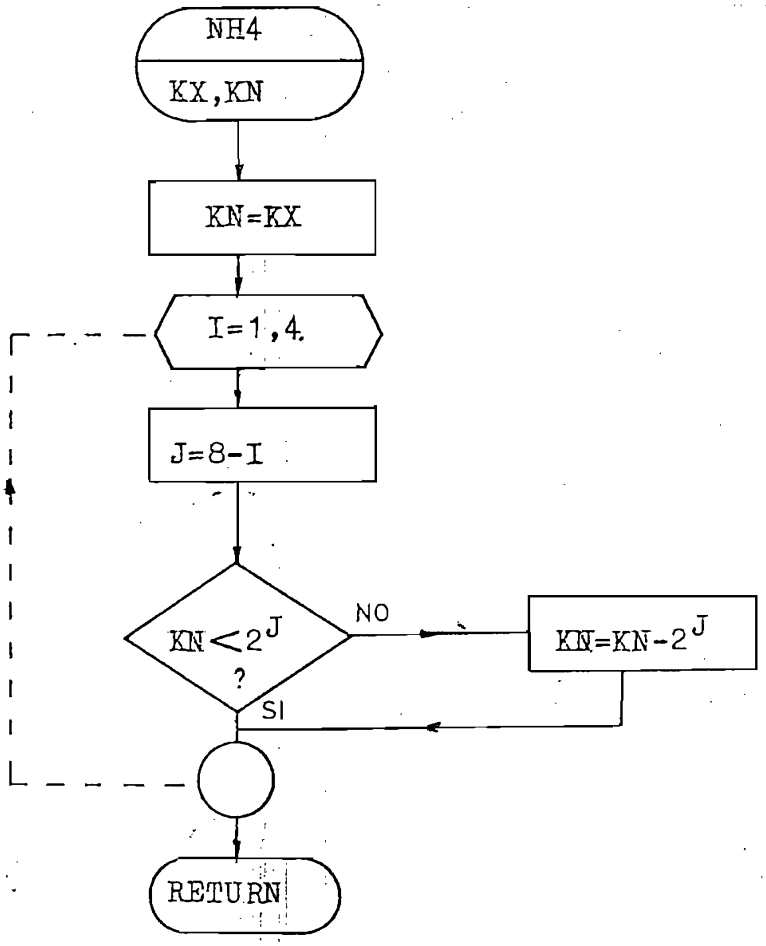


FIG.4-4-56

C A P I T U L O 5

VARIOS EJEMPLOS DE APLICACION USANDO EL SIMULADOR IMPLEMENTADO . RESULTADOS Y EVALUACION DE LOS MISMOS

En este capítulo se diseñarán varios ejemplos para probar el Simulador,para esto,en base a lo que se dijo en el punto 3-6 ,hay que hacer una planificación de los experimentos a realizar,esto se refiere,a escoger el conjunto de estos y especificar la manera como trabaja cada experimento.

Los ejemplos,serán programas simulados que utilizan las instrucciones más representativas ; la ejecución de estos puede ser realizada en algunas formas como son : ejecución de pasos de programa,ejecucción de unlnúmero de instrucciones,rastreo de pasos o de saltos.

En estos ejemplos se tendrá la posibilidad de usar algunos de los comandos disponibles , así como tambíen los macrocomandos (todos estos fueron explicados en 3-4).

A continuación se explican los programas simulados que se han implementado y que previamente se hán ensamblado.

en el Cross Assembler del Microprocesador M6800.(El manejo del Cross Assembler esta explicado en la Tesis del mismo indicada em la Bibliografía).

En el primer ejemplo,se hace un listado de los macrocomandos gravados;estos,serán llamados en cada uno de los ejemplos.siguientes.

EJEMPLO 5-1

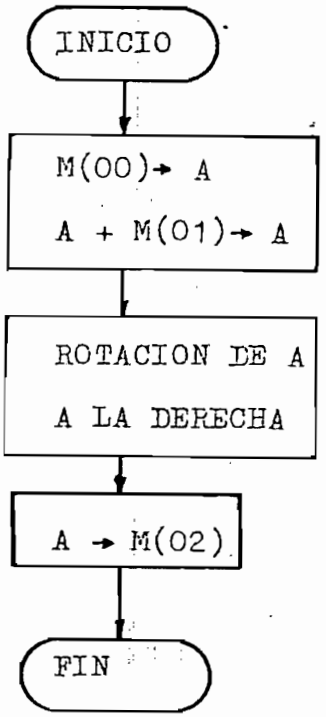
-Enunciado.-Programa que obtiene el promedio de dos números binarios de 8 bits,los mismos que se encuentran almacenados en las localidades : A = $0000 y B = $0001 ; el resultado se almacena en la localidad $0003 . Comienza el programa en la localidad $0100 . (El signo:$ indica que el número a continuación es hexadecimal)

-Algoritmo.-Se suma los dos números y se divide para dos,mediante una rotación a la derecha.

-Mapa de Memoria.-Sí N1 : primer número y N2 : 2do.número se tendría así:

0000	N1
0001	N2
0002	(N1 + N2)/2

-Diagrama de Flujo .-



```
EDITO

WHAT FILE  4?>
PROM

      { CARGA INICIAL DE PROGRAMA? (S/N) }

      INSERTANDO LINEAS? (S/N)


      ELIMINANDO LINEAS? (S/N)


      LISTAR UN NUMERO DE LINEAS? (S/N)


      LISTAR EL PROGRAMA? (S/N)
S
```

```
    1:      00001                           NAM   PROM
    2:      00002 0100                      ORG   $0100
    3:      00003                  *PROMEDIO DE DOS NUMEROS QUE ESTAN EN:00 Y 01
    4:      00004                  *RESULTADO EN 02
    5:      00005 0100 96 00                LDAA  $00
    6:      00006 0102 9B 01                ADDA  $01
    7:      00007 0104 44                   LSRA
    8:      00008 0105 97 02                STAA  $02
    9:      00009 0107 3F                   SWI
   10:      00010                           END

    LISTAR EL PROGRAMA? (S/N)


    MODIFICAR UN REGISTRO ? (S/N)


    FIN DEL TRABAJO? (S/N)
S

==>
```

```
WHAT FILE  1?>
MEMRO

WHAT FILE  2?>
CO2P

WHAT FILE  3?>
REGIS

WHAT FILE  4?>
PROM

WHAT FILE  5?>
COM

WHAT FILE  6?>
MACRO

WHAT FILE  7?>
ERR

WHAT FILE  8?>
NMONI
```

```
+-----------------------------------------------------+
|          ESCUELA   POLITECNICA   NACIONAL           |
|          FACULTAD DE INGENIERIA ELECTRICA           |
|          TESIS:SIMULADOR DEL MICROPROCESADOR        |
|                        M6800                        |
|          REALIZADA POR:EDISON ZUNIGA HIDALGO        |
|          DIRECTOR DE TESIS:ING.EDGAR TORRES P.      |
|          FECHA:ABRIL 1984    QUITO-ECUADOR          |
+-----------------------------------------------------+
```

```
?
CP

  UN PROGRAMA DE : 10LINEAS
DESDE LA LOCALIDAD :0100(HEXADECIMAL) FUE CARGADO
?
PM 0000,58,4F,00
?
VM 0000,03
  DIRECCION          CONTENIDO
  0000               58
  0001               4F
  0002               00
?
```

```
LM
HXD (LC.CO 00.EJ 00A0,00FF.VM 0001,03.NL)
TRAI (PM 0001,00,00,00,00,00,00,00,00,00,00.VM 0001,0A)
TRA1 (LC.VM E001,0A.CO 00.EJ 0018,0FFF.VM 0001,0A.NL)
PAR1 (PM 00A1,00,00,00,00,00,00,00,00,00,00.VM 00A1,0A)
PAR2 (LC.VM 0001,0A.CO 00.EJ 0050,0FFF.VM 00A1,0A.NL)
P01 (LC.CO 01.RN 0100,00FF.VM 0000,03.NL)
SUM1 (LC.CO 01.RS 00B0,00FF.VM 0021,0C.NL)
MAYN (LC.CO 00.EJ 0B00,00FF.VM 0000,05.NL)
ORD1 (LC.VM 00A1,0A.CO 00.EJ 00C0,0FFF.VM 00A1,0A.NL)
TBA1 (LC.CO 00.EJ 0700,0FFF.VM 0000,0A)
```

```
PO1

CO 01


RN 0100,00FF


    I      O      P      E      X      A      B      HINZVC    S          T
  0100   LDAA   0100   0100   0000   68     66     000100    AA80         0
  0100   LDAA   0102   0000   0000   58     66     000000    AA80         3
  0102   ADDA   0104   0001   0000   A7     66     101010    AA80         6
  0104   LSRA   0105   0104   0000   53     66     100011    AA80         8

    I      O      P      E      X      A      B      HINZVC    S          T
  0105   STAA   0107   0002   0000   53     66     100001    AA80        12
  0107   SWI    0108   0107   0000   53     66     100001    AA79        24

VM 0000,03

  DIRECCION            CONTENIDO
  0000                 58
  0001                 4F
  0002                 53

NL

?
PM 0000,29,A6,00
?
PO1

CO 01


RN 0100,00FF


    I      O      P      E      X      A      B      HINZVC    S          T
  0100   LDAA   0100   0100   0000   53     66     100001    AA79         0
  0100   LDAA   0102   0000   0000   29     66     100001    AA79         3
  0102   ADDA   0104   0001   0000   CF     66     001000    AA79         6
  0104   LSRA   0105   0104   0000   67     66     000011    AA79         8

    I      O      P      E      X      A      B      HINZVC    S          T
  0105   STAA   0107   0002   0000   67     66     000001    AA79        12
  0107   SWI    0108   0107   0000   67     66     000001    AA72        24

VM 0000,03

  DIRECCION            CONTENIDO
  0000                 29
  0001                 A6
  0002                 67

NL

?
/*

==>
```

-Evaluación de resultados .-Al terminar la ejecución del programa se puede revisar el contenido de la memoria de dirección 0002 donde se almacena el resultado,por medio del comando VM; este corresponde al promedio de los dos números ingresados; al inicio se puede poner en el registro A el valor de 00 con el comando PR :.

Para este programa se ha conformado un macrocomando llamado P01,que incluye el comando RN que sirve para rastrear el programa,es decir, para ir observando los cambios en los registros luego de la ejecución de cada instrucción.

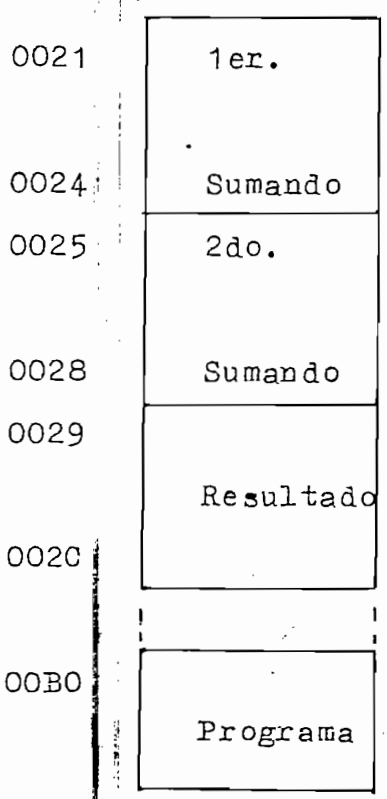
EJEMPLO 5-2

-Enunciado .-Programa para suma decimal de 4 bytes (8 sumandos) que están localizados así : 1er. sumando : 0021 - 0024 ; 2do. sumando : 0025 - 0028 ; resultado : 0029 -002C ; el programa comienza en : 00B0

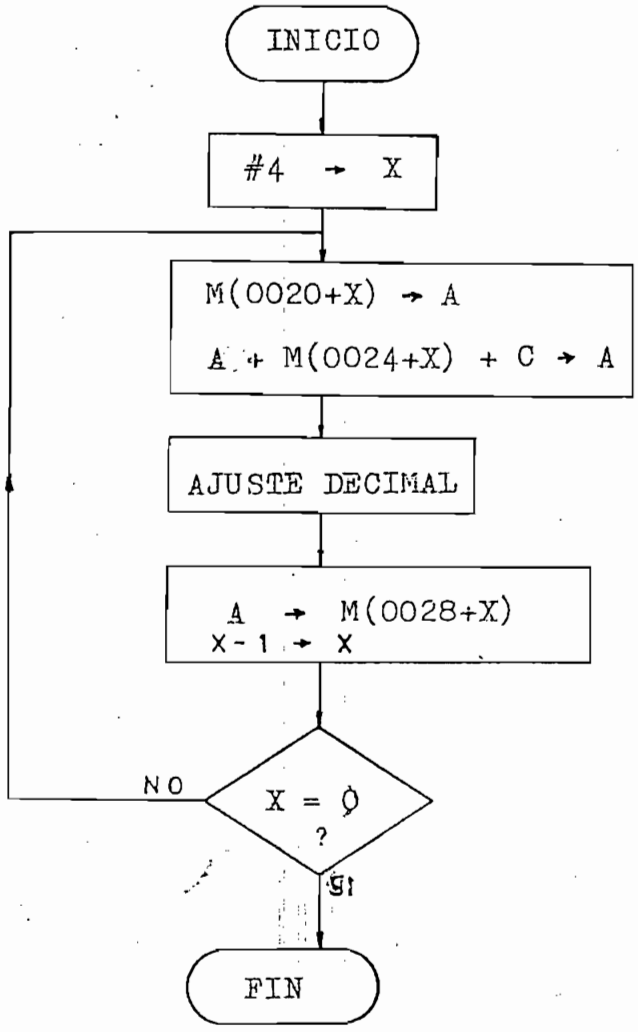
-Algoritmo .-Se suma byte por byte ,considerando el carry.

					C
1er sumando	A_4	A_3	A_2	A_1	
2do.sumando	B_4	B_3	B_2	B_1	
Resultado	R_4	R_3	R_2	R_1	

-Mapa de Memoria .-



-Diagrama de Flujo .-



```
/*

==>
EDITO

WHAT FILE  4?>
SUN

      { CARGA INICIAL DE PROGRAMA? (S/N) }

      INSERTANDO LINEAS? (S/N)


      ELIMINANDO LINEAS? (S/N)


      LISTAR UN NUMERO DE LINEAS? (S/N)


      LISTAR EL PROGRAMA? (S/N)
S
```

```
 1:      00001                          NAM    SUN
 2:      00002 00B0                     ORG    $00B0
 3:      00003                        *PROGRAMA PARA SUMA DECIMAL DE 2 NUMEROS
 4:      00004                        *DE 4 BYTES.POSICION DEL BYTE MAS
 5:      00005                        *SIGNIFICATIVO DEL 1ER.SUMANDO :$21 ;
 6:      00006                        *DEL 2DO.SUMANDO :$25 ; DEL RESULTADO:$29
 7:      00007 00B0 0C                  CLC
 8:      00008 00B1 CE 0004             LDX    #$0004
 9:      00009 00B4 A6 20    LAZO       LDAA   $20,X
10:      00010 00B6 A9 24               ADCA   $24,X
11:      00011 00B8 19                  DAA
12:      00012 00B9 A7 28               STAA   $28,X
13:      00013 00BB 09                  DEX
14:      00014 00BC 26 F6               BNE    LAZO
15:      00015 00BE 3F                  SWI
16:      00016                          END
```

LISTAR EL PROGRAMA? (S/N)

```
S6800

WHAT FILE  1?>
MEMRO

WHAT FILE  2?>
CO2P

WHAT FILE  3?>
REGIS

WHAT FILE  4?>
SUN

WHAT FILE  5?>
COM

WHAT FILE  6?>
MACRO

WHAT FILE  7?>
ERR

WHAT FILE  8?>
NMONI
```

```
+-------------------------------------------------------+
|          ESCUELA   POLITECNICA   NACIONAL             |
|          FACULTAD DE INGENIERIA ELECTRICA             |
|          TESIS:SIMULADOR DEL MICROPROCESADOR          |
|                      M6800                            |
|          REALIZADA POR:EDISON ZUNIGA HIDALGO          |
|          DIRECTOR DE TESIS:ING.EDGAR TORRES P.        |
|          FECHA:ABRIL-1984    QUITO-ECUADOR            |
+-------------------------------------------------------+
```

```
?
CP

CP


  UN PROGRAMA DE : 16LINEAS
DESDE LA LOCALIDAD :00B0(HEXADECIMAL) FUE CARGADO
?
NL

NL

?
PM 0021,50,16,25,34,18,15,39,43,00,00,00,00
?
```

SUM1

CO 01

RS 00B0,00FF

I	O	P	E	X	A	B	HINZVC	S	T
00BC	BNE	00B4	00BD	0003	77	66	000000	AA87	31
00BC	BNE	00B4	00BD	0002	64	66	000000	AA87	57
00BC	BNE	00B4	00BD	0001	31	66	000000	AA87	83
00BC	BNE	00BE	00BD	0000	68	66	000100	AA87	109
00BE	SWI	00BF	00BE	0000	68	66	000100	AA80	121

VM 0021,0C

DIRECCION	CONTENIDO
0021	50
0022	16
0023	25
0024	34
0025	18
0026	15
0027	39
0028	43
0029	68
002A	31
002B	64
002C	77

NL

?

-Evaluación de Resultados.-

Se puede visualizar los resultados luego de la ejecución del programa,para esto los comandos respectivos han sido definidos en un macro de nombre SUM1.

La forma de ejecución del programa se hace por medio del comando RS ,el cual presenta el cambio en los registros,cada vez que hay un salto en el programa simulado.

Para revisar los macros creados,se utiliza el comando LM.

EJEMPLO 5-3

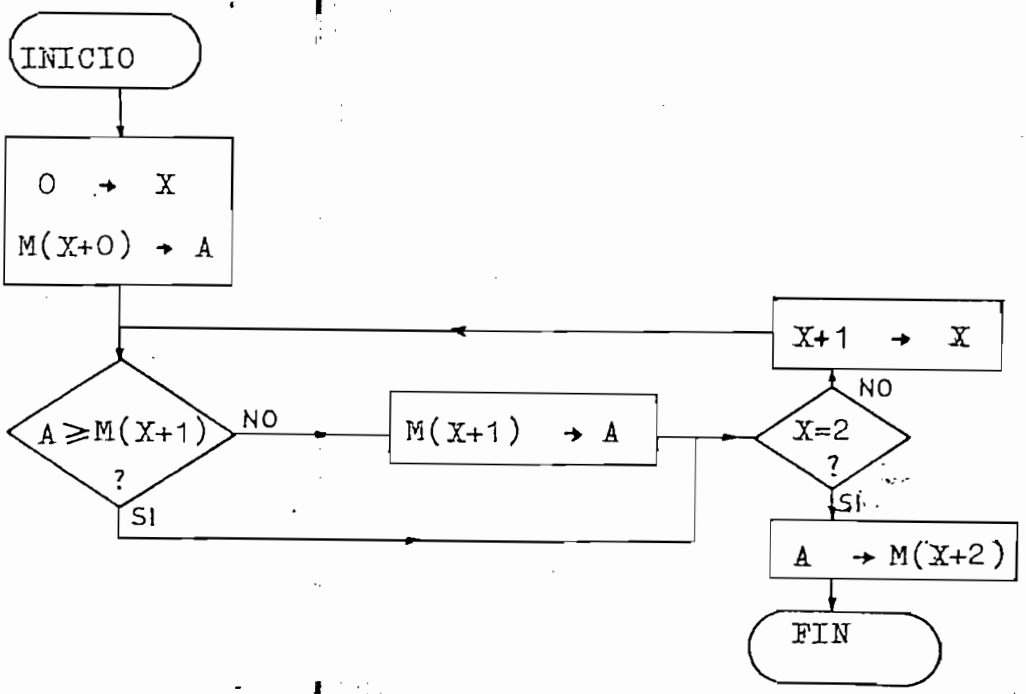
-Enunciado .-Programa que identifica el mayor de 4 números binarios de 8 bits,que se encuentran almacenados en las localidades 0000,0001,0002,0003 y lo guarda en la localidad 0004 .Comienza el programa en la localidad CB00.

-Algoritmo .-El programa va comparando dato por dato y reteniendo el mayor de ellos.

-Mapa de Memoria .- Sí N1,N2,N3,N4 son los números,se tendría:

0000	N1
0001	N2
0002	N3
0003	N4
0004	N mayor

-Diagrama de Flujo .-



```
 1:      00001                          NAM    MAY4
 2:      00002 0B00                     ORG    $0B00
 3:      00003                *MAYOR DE 4 NUMEROS QUE ESTAN EN
 4:      00004                *LOCALIDADES :00,01,02,03 ; EL MAYOR
 5:      00005                *SE ALMACENA EN LA LOCALIDAD 04
 6:      00006 0B00 CE 0000             LDX    #$0000
 7:      00007 0B03 A6 00               LDAA   $00,X     SE CARGA EL 1ER NU
 8:      00008 0B05 A1 01    CONC       CMPA   $01,X
 9:      00009 0B07 2C 02               BGE    COMX
10:      00010 0B09 A6 01               LDAA   $01,X     EN A SE CARGA EL M
11:      00011 0B0B 8C 0002  COMX       CPX    #$0002    PRUEBA SI ESTAN TO
12:      00012 0B0E 27 03               BEQ    TERM        LOS  DATOS
13:      00013 0B10 08                  INX
14:      00014 0B11 20 F2               BRA    CONC
15:      00015 0B13 A7 02    TERM       STAA   $02,X
16:      00016 0B15 3F                  SWI
17:      00017                          END

   LISTAR EL PROGRAMA? (S/N)


   MODIFICAR UN REGISTRO ? (S/N)


   FIN DEL TRABAJO? (S/N)
S

==>
```

```
S6800

WHAT FILE  1?>
MEMRO

WHAT FILE  2?>
CO2P

WHAT FILE  3?>
REGIS

WHAT FILE  4?>
MAY4

WHAT FILE  5?>
COM

WHAT FILE  6?>
MACRO

WHAT FILE  7?>
ERR

WHAT FILE  8?>
NMONI
```

```
+-----------------------------------------------------+
|          ESCUELA   POLITECNICA   NACIONAL           |
|          FACULTAD DE INGENIERIA ELECTRICA           |
|          TESIS:SIMULADOR DEL MICROPROCESADOR        |
|                        M6800                        |
|          REALIZADA POR:EDISON ZUNIGA HIDALGO        |
|          DIRECTOR DE TESIS:ING.EDGAR TORRES P.      |
|          FECHA:ABRIL-1984    QUITO-ECUADOR          |
+-----------------------------------------------------+
```

```
?
CP

CP


  UN PROGRAMA DE : 17LINEAS
DESDE LA LOCALIDAD :0B00(HEXADECIMAL) FUE CARGADO
?
NL

NL

?
```

```
PM 0000,50,60,10,55,00
?
MAYN

CO 00


EJ 0B00,00FF

    I      O      P      E      X      A      B      HINZVC   S          T
   0B15   3F     0B16   0B15   0002   60     00     000000   AA80       95

VM 0000,05

  DIRECCION          CONTENIDO
  0000               50
  0001               60
  0002               10
  0003               55
  0004               60

NL

?
PM 0000,26,4F,5A,6A,00
?
MAYN

CO 00


EJ 0B00,00FF

    I      O      P      E      X      A      B      HINZVC   S          T
   0B15   3F     0B16   0B15   0002   6A     00     000001   AA79       105

VM 0000,05

  DIRECCION          CONTENIDO
  0000               26
  0001               4F
  0002               5A
  0003               6A
  0004               6A

NL

?
/*

==>
```

Evaluación de Resultados.- Para la Simulación del Programa se han colocado primeramente los datos,por medio del comando PM.

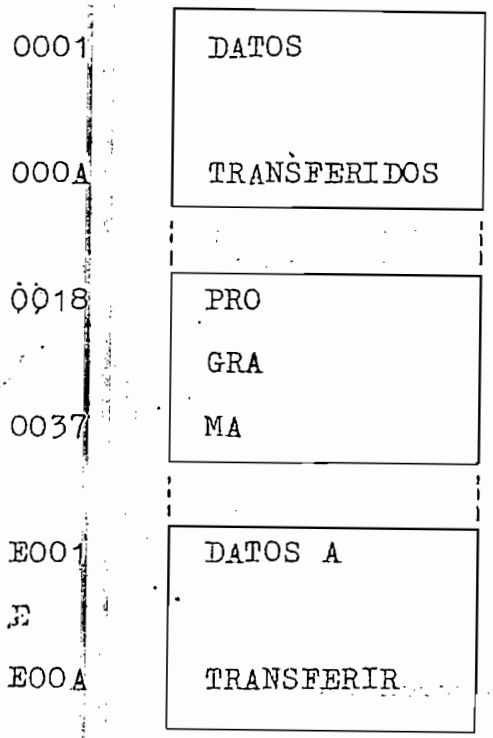
Para la ejecución del programa y la visualización de resultados,se conforma un macrocomando MAYN que contiene los comandos EJ (Ejecutar un programa) y VM (Visualizar memoria).

EJEMPLO 5-4

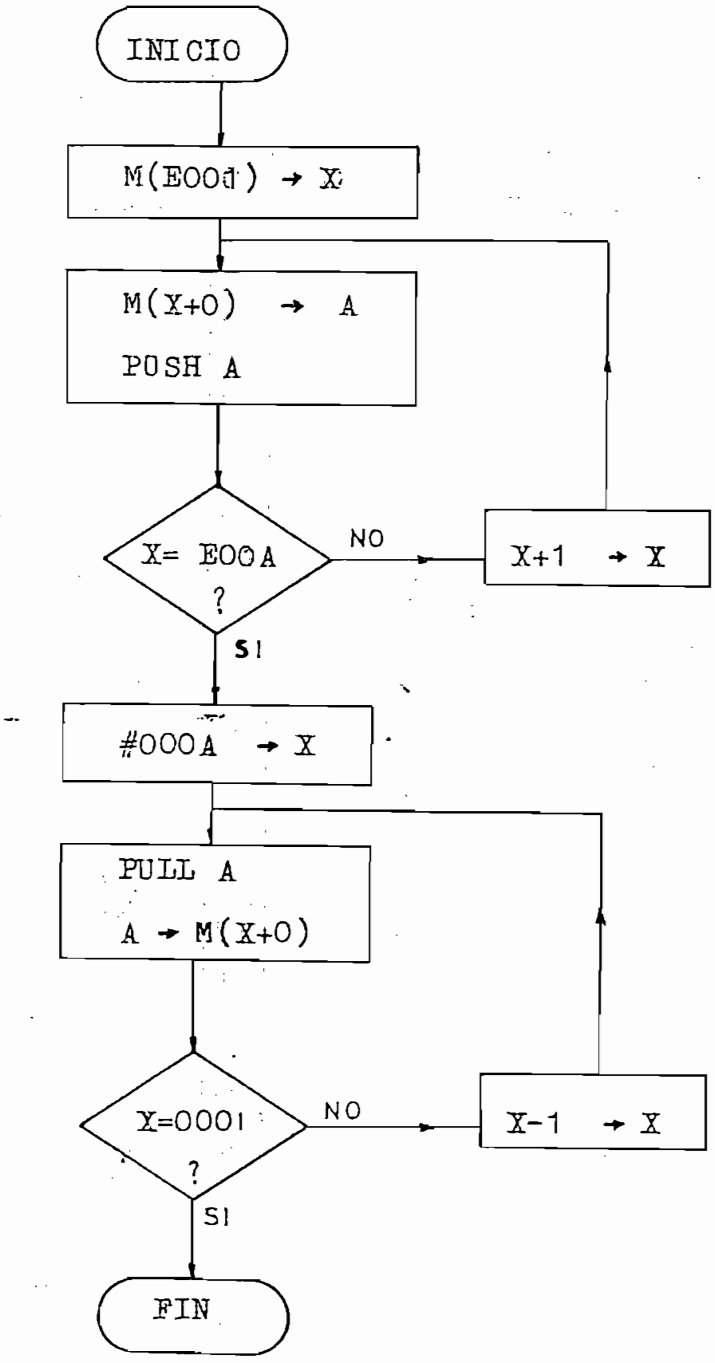
-Enunciado .-Transferir 10 bytes de un lugar de memoria a otro,utilizando el'stack de memoria'.

--Algoritmo .-Se va tomando dato por dato y colocando en el stack,para luego de que estén todos los datos almacenados ahí,sean sacados y puestos en otro lugar de memoria.

-Mapa de Memoria .-



-Diagrama de Flujo .-



```
    1:      00001                           NAM    TR10
    2:      00002 0018                      ORG    $0018
    3:      00003                  *TRANSFERENCIA EN MEMORIA DE 10 BYTES
    4:      00004                  *DESDE $E001-$E00A HASTA $01-$0A
    5:      00005 0018 8E AAAA              LDS    #$AAAA
    6:      00006 001B CE E001              LDX    #$E001
    7:      00007 001E A6 00    PON         LDAA   00,X
    8:      00008 0020 36                   PSHA                CARGA DATOS EN STAC
    9:      00009 0021 8C E00A              CPX    #$E00A
   10:      00010 0024 27 08                BEQ    SAC
   11:      00011 0026 08                   INX
   12:      00012 0027 20 F5                BRA    PON
   13:      00013 0029 CE 000A  SAC         LDX    #$000A
   14:      00014 002C 32       OBT         PULA                SACA DATOS DEL STAC
   15:      00015 002D A7 00                STAA   00,X         GRAVA DATOS EN $01-
   16:      00016 002F 8C 0001              CPX    #$0001
   17:      00017 0032 27 03                BEQ    TERM
   18:      00018 0034 09                   DEX
   19:      00019 0035 20 F5                BRA    OBT
   20:      00020 0037 3F       TERM        SWI
   21:      00021                           END

    LISTAR EL PROGRAMA? (S/N)

    MODIFICAR UN REGISTRO ? (S/N)

    FIN DEL TRABAJO? (S/N)
S

==>
```

```
S6800

WHAT FILE  1?>
MEMRO

WHAT FILE  2?>
CO2P

WHAT FILE  3?>
REGIS

WHAT FILE  4?>
TR10

WHAT FILE  5?>
COM

WHAT FILE  6?>
MACRO

WHAT FILE  7?>
ERR

WHAT FILE  8?>
NMONI
```

```
+-----------------------------------------------------------+
|          ESCUELA   POLITECNICA   NACIONAL                 |
|          FACULTAD DE INGENIERIA ELECTRICA                 |
|          TESIS:SIMULADOR DEL MICROPROCESADOR              |
|                      M6800                                |
|          REALIZADA POR:EDISON ZUNIGA HIDALGO              |
|          DIRECTOR DE TESIS:ING.EDGAR TORRES P.            |
|          FECHA:ABRIL-1984    QUITO-ECUADOR                |
+-----------------------------------------------------------+
```

```
?
CP

  UN PROGRAMA DE : 21LINEAS
DESDE LA LOCALIDAD :0018(HEXADECIMAL) FUE CARGADO
?
```

```
TRA1
   DIRECCION              CONTENIDO
   0001                   00
   0002                   00
   0003                   00
   0004                   00
   0005                   00
   0006                   00
   0007                   00
   0008                   00
   0009                   00
   000A                   00
?
TRA1

VM E001,0A

   DIRECCION              CONTENIDO
   E001                   11
   E002                   66
   E003                   44
   E004                   77
   E005                   20
   E006                   25
   E007                   60
   E008                   51
   E009                   10
   E00A                   30

CO 00


EJ 0018,0FFF

     I      O      P      E      X      A      B      HINZVC   S          T
    0037   3F     0038   0037   0001   11     00     000101   AAA3       495

VM 0001,0A

   DIRECCION              CONTENIDO
   0001                   11
   0002                   66
   0003                   44
   0004                   77
   0005                   20
   0006                   25
   0007                   60
   0008                   51
   0009                   10
   000A                   30

NL

?
/*

==>
```

-Evaluación de Resultados .-Los datos a ser transferidos están a partir de la localidad E001.

La ejecución del programa se realiza utilizando un macrocomando de nombre TRA1,el cual además contiene un comando de visualizar los datos transferidos.

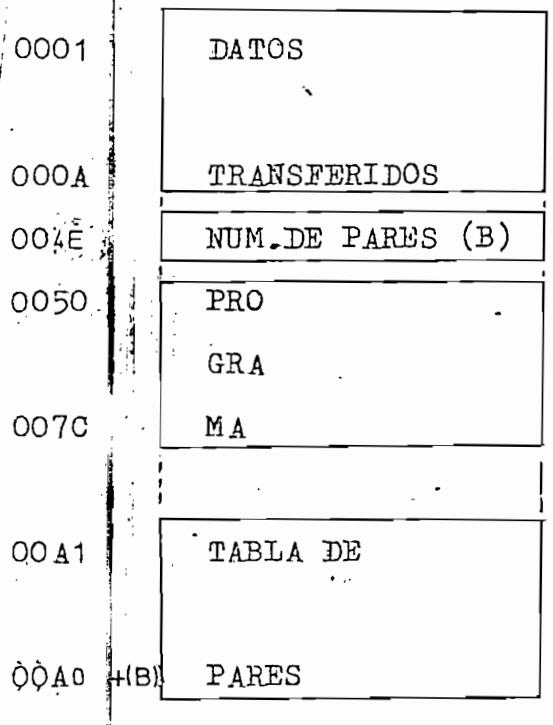
Al inicio,hay que poner ceros en las localidades de memoria donde van a estar los datos transferidos,para esto se conforma un macrocomando llamado TRAI.

EJEMPLO 5-5

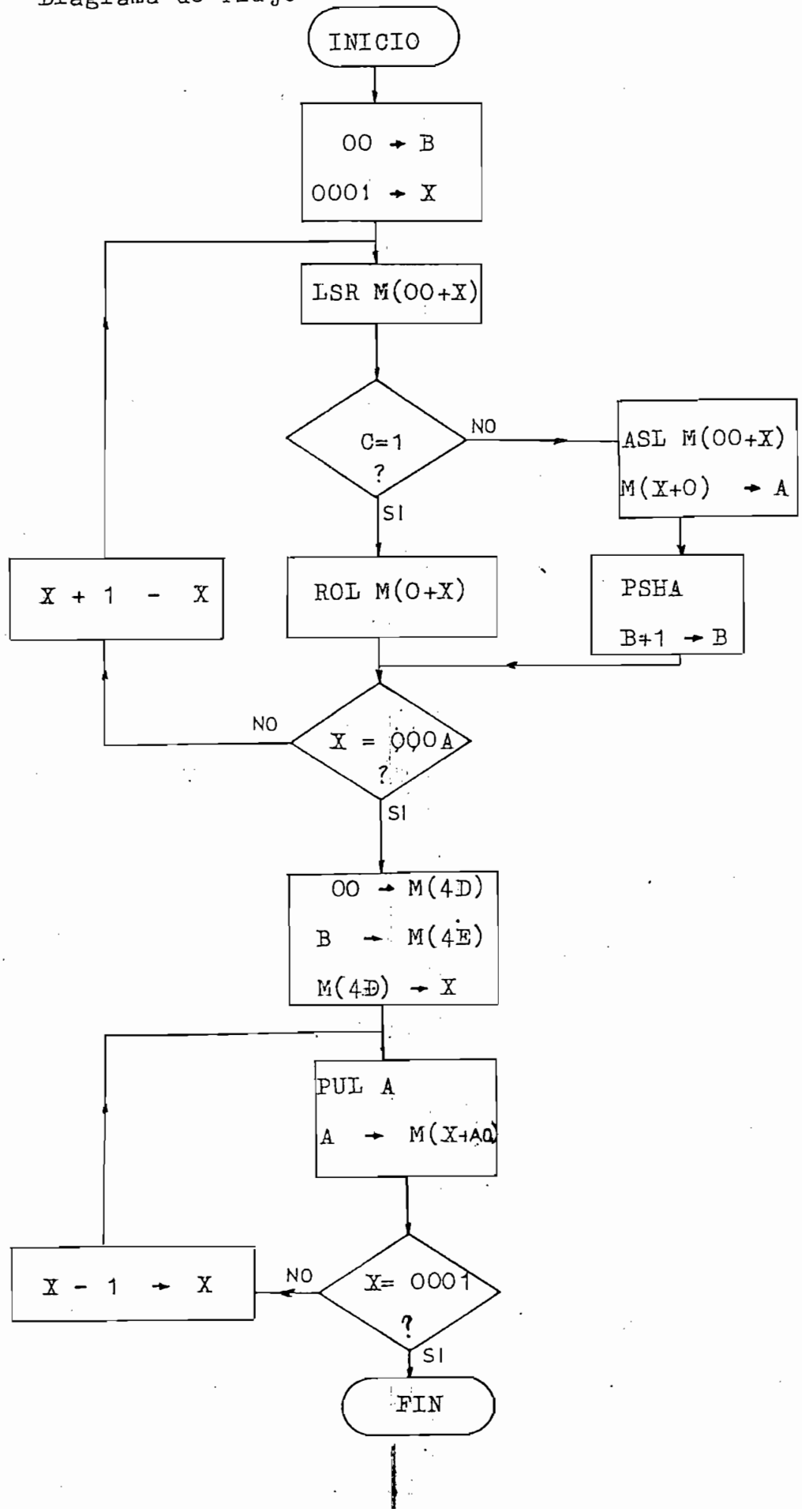
-Enunciado .- Busqueda de números pares en la tabla de valores transferidos por el programa del ejemplo 5-5.

-Algoritmo .- Para analizar cuales son los números pares, se utiliza una instrucción de rotación hacia la derecha; y los que sean, son guardados en el stack de memoria, para posteriormente ser puestos a partir de la localidad 00A1.

-Mapa de Memoria.-



-Diagrama de Flujo



```
 1:      00001                           NAM   PARS
 2:      00002 0050                      ORG   $0050
 3:      00003                  *BUSQUEDA DE NUMEROS PARES DE UNA TABLA
 4:      00004                  *QUE ESTAN ENTRE $01 A $0A
 5:      00005                  *NUMEROS PARES SE GUARDAN DESDE $A1
 6:      00006                  *NUMERO DE VALORES PARES ESTA EN $4E
 7:      00007 0050 5F                   CLRB
 8:      00008 0051 CE 0001              LDX   #$0001
 9:      00009 0054 64 00       SAL1     LSR   00,X      DESPLAZA EL DATO
10:      00010 0056 25 08                BCS   SAL2      SALTA SI ES IMPAR
11:      00011 0058 68 00                ASL   00,X
12:      00012 005A A6 00                LDAA  00,X
13:      00013 005C 36                   PSHA            GUARDA VALORES PARE
14:      00014 005D 5C                   INCB            AUMENTA NUMERO DE P
15:      00015 005E 20 02                BRA   SAL3
16:      00016 0060 69 00       SAL2     ROL   00,X
17:      00017 0062 8C 000A     SAL3     CPX   #$000A
18:      00018 0065 27 03                BEQ   SAL4
19:      00019 0067 08                   INX
20:      00020 0068 20 EA                BRA   SAL1
21:      00021 006A D7 4E       SAL4     STAB  $4E
22:      00022 006C 7F 004D              CLR   $4D
23:      00023 006F DE 4D                LDX   $4D
24:      00024 0071 32          SAL5     PULA            SACA VALORES PARES
25:      00025 0072 A7 A0                STAA  $A0,X     ALMACENA VALORES PA
26:      00026 0074 8C 0001              CPX   #$0001
27:      00027 0077 27 03                BEQ   SAL6
28:      00028 0079 09                   DEX
29:      00029 007A 20 F5                BRA   SAL5
30:      00030 007C 3F          SAL6     SWI
31:      00031                           END

    LISTAR EL PROGRAMA? (S/N)


    MODIFICAR UN REGISTRO ? (S/N)


    FIN DEL TRABAJO? (S/N)
S

==>
```

```
S6800

WHAT FILE  1?>
MEMRO

WHAT FILE  2?>
CO2P

WHAT FILE  3?>
REGIS

WHAT FILE  4?>
PARS

WHAT FILE  5?>
COM

WHAT FILE  6?>
MACRO

WHAT FILE  7?>
ERR

WHAT FILE  8?>
NMONI
```

```
+-------------------------------------------------------+
|          ESCUELA   POLITECNICA   NACIONAL             |
|          FACULTAD DE INGENIERIA ELECTRICA             |
|          TESIS:SIMULADOR DEL MICROPROCESADOR          |
|                      M6800                            |
|          REALIZADA POR:EDISON ZUNIGA HIDALGO          |
|          DIRECTOR DE TESIS:ING.EDGAR TORRES P.        |
|          FECHA:ABRIL-1984    QUITO-ECUADOR            |
+-------------------------------------------------------+
```

```
?
CP

  UN PROGRAMA DE : 31LINEAS
DESDE LA LOCALIDAD :0050(HEXADECIMAL) FUE CARGADO
?
```

```
PAR1
   DIRECCION            CONTENIDO
   00A1                 00
   00A2                 00
   00A3                 00
   00A4                 00
   00A5                 00
   00A6                 00
   00A7                 00
   00A8                 00
   00A9                 00
   00AA                 00
?
PAR2

VM 0001,0A

   DIRECCION            CONTENIDO
   0001                 11
   0002                 66
   0003                 44
   0004                 77
   0005                 20
   0006                 25
   0007                 60
   0008                 51
   0009                 10
   000A                 30

CO 00


EJ 0050,0FFF

    I      O      P      E      X      A      B      HINZVC    S          T
   007C   3F     007D   007C   0001   66     06     000100    AA9C       585

VM 00A1,0A

   DIRECCION            CONTENIDO
   00A1                 66
   00A2                 44
   00A3                 20
   00A4                 60
   00A5                 10
   00A6                 30
   00A7                 00
   00A8                 00
   00A9                 00
   00AA                 00

NL

?
/*

==>
```

-Evaluación de Resultados .-

Las localidades en las cuales irán los números pares som inicializadas com cero y visualizadas ,por medio de un macrocomando llamado PAR1 ;esto se hace antes de la ejecución del programa.

Para la ejecución del programa y la comprobación de los resultados,se conforma el macrocomando PAR2 ; este contiene los comandos LC y CO para visualizar los comandos que se ejecutan y escoger el código objeto para la salida; y también comtiene los comandos EJ y VM para la ejecución del programa y la presentación de resultados.

C A P I T U L O 6

CONCLUSIONES Y RECOMENDACIONES

El Simulador del Microprocesador M6800 representa una ayuda eficaz en la formación de programas realizados por este; pues utilizándose el Simulador conjuntamente con el Cross Assembler hay facilidad en la implementación de programas, en la depuración de los mismos, en la comprobación de programas para distintos valores; como se ha podido observar en los ejemplos realizados.

El Simulador tendría distintas aplicaciones desde el campo pedagógico en el aprendizaje del Microprocesador M6800, su programación, sus características; hasta el campo de aplicaciones prácticas en la conformación de programas para sistemas físicos que utilizan este microprocesador.

El Simulador combinado con el Cross Assambler son herramientas excelentes para el diseño y desarrollo de Sistemas basados en microprocesadores, en este caso el M6800.

Este Sistema Simulador a partir del código objeto generado por el Cross Assambler, ha permitido realizar co

rridas, pruebas simuladas y análisis de los diferentes registros y áreas de memoria; con el fin de determinar si el programa o subrutina ha cumplido con su objetivo.

Para la implementación del Simulador se ha utilizado el lenguaje Fortran, siendo el de mayor difusión y el que es generalmente requerido para aplicaciones científicas por sus características de tener sentencias que resultan similares a las expresiones matemáticas, por su interdependencia con el diagrama de flujo, etc. Todo esto se ha podido comprobar a lo largo de la programación del Sistema Simulador realizado.

La utilización de los conceptos de máquinas de estado finito en cierta parte de la programación del Simulador; se hizo necesaria, para que el manejo de memoria se facilite en buena medida.

El hecho de haberse utilizado archivos, ha permitido tener almacenada información permanente y es así que se han organizado 8 archivos; que guardan las instrucciones, los macrocomandos, los comandos, los errores, etc.

El empleo de los llamados macrocomandos en el manejo del Simulador ha generado múltiples ventajas, en tiempo y en la distribución de comandos a utilizarse; esto puede comprobarse en los ejemplos desarrollados.

Es importante indicar la limitación que tendría la Simulación de un Sistema, es así que en el caso del M6800, se ha hecho circunscribiendo la Simulación a los registros,

la memoria que puede direccionar, el decodificador de instrucciones, la Unidad Aritmética y Lógica; y se ha dejado como ambiente del Sistema, es decir, lo que no se simula, las conexiones exteriores, señales de control de tiempo, señales de control del bus de datos, señales de control del bus de direcciones.

En la Simulación, el crear modelos ha sido de mucha utilidad, ya que en estos se recoge la información del Sistema y es un paso indispensable para ir simulando de una manera segura y ordenada sin olvidar todas las características del Sistema.

La organización de Sistema en bloques ha permitido visualizar de manera global la conformación y el funcionamiento del mismo; pudiendo reconocer además cada bloque como un Subsistema, enfocándoselos de forma individual, lo que ha hecho manejable al Sistema.

Los comandos realizados, dan lugar a diversas formas de operar el Simulador, así por ejemplo para la ejecución de un programa hay 4 formas diferentes de hacerlo; para el control de formatos e impresión de salidas hay 7 comandos, para poner valores existen 4 comandos; para visualizar valores se han implementado 5 comandos; para simulación de interrupción luego de instrucción de espera de interrupción, hay 2 comandos, y comandos a tabla de macro comandos hay 2.

La documentación de los programas es suficiente in

formación de los mismos, pudiendo servir en lo posterior para el mantenimiento de los programas, para ampliación a nuevas características que se desee, o simplemente para analizar la forma en que se han realizado.

La detección de errores, por parte del Simulador, de:comandos no ingresados correctamente, macrocomandos mal definidos, etc, ofrecerá al usuario una rápida identificación de los mismos y la inmediata corrección, dando continuidad en el manejo del Simulador.

Es recomendable siempre referirse al Manual del Usuario para asegurarse de que la forma de manejo del Simulador sea la correcta, especialmente cuando se ha recibido un mensaje de error.

En base a la observación del trabajo del Simulador en la ejecución de programas, se recomienda utilizar la opción de salida en código objeto que implica un menor tiempo; pero si se desea rastrear un programa es más conveniente utilizar salida en código nmónico para visualización de las instrucciones que se van realizando, a pesar de que esta última forma ocupa mayor tiempo.

Es indispensable indicar aquí, que la ejecución de un programa Simulado toma mayor tiempo que si sería hecho en el sistema físico; es por esto que, se recomienda realizar programas que abarquen pocos lazos en el mismo; ya que por ser una simulación no se requiere la comprobación total de programas, sino más bien un análisis de las partes más importantes de un programa, por ejemplo:corridas par-

ciales en ciertos programas.

Con el Simulador se ha querido aportar en cierta medida, en la programación de las aplicaciones que se realizan con el microprocesador M6800; facilitándose el desarrollo de los mismos, por que sus programas pueden ser probados y depurados a satisfacción del usuario.

A P E N D I C E S

APENDICE A :LISTADOS DE PROGRAMAS

```
T 96800
96800.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

      DEFINE FILE 1(66000,2,U,J1)
      DEFINE FILE 2(256,10,U,J2)
      DEFINE FILE 3(1,20,U,J3)
      DEFINE FILE 4(1000,160,U,J4)
      DEFINE FILE 5(22,12,U,J5)
      DEFINE FILE 6(30,150,U,J6)
      DEFINE FILE 7(50,160,U,J7)
      DEFINE FILE 8(400,32,U,J8)
      DIMENSION LL(72),L5(5),KCOM(2),NPOC(2),NPUN(10)
      INTEGER BLANCO/' '/
      WRITE(1,6)
6     FORMAT(/,10X,52('-'),/,10X,'|',10X,'ESCUELA  POLITECNICA  '
     1,'NACIONAL',8X,'|',/,10X,'|',10X,'FACULTAD DE INGENIERIA ',
     1'ELECTRICA',8X,'|',/,10X,'|',10X,'TESIS:SIMULADOR DEL MICRO',
     1'PROCESADOR',5X,'|',/,10X,'|',25X,'M6800',20X,'|',/,10X,
     1'|',10X,'REALIZADA POR:EDISON ZUNIGA HIDALGO',5X,'|',/,10X,
     1'|',10X,'DIRECTOR DE TESIS:ING.EDGAR TORRES P.',3X,'|',/,10X,
     1'|',10X,'FECHA:ABRIL-1984',4X,'QUITO-ECUADOR',7X,'|',/,10X,
     152('-'))
      INB=16
      CALL DEK(INB,M,K)
      DO 5 JJ5=2,3
      READ(5'JJ5)(KCOM(I),I=1,2),NSEC
5     WRITE(5'JJ5)(KCOM(I),I=1,2),NSEC,INB,M,K
      JBPM=0
      READ(5'13)(KCOM(I),I=1,2),NSEC
      WRITE(5'13)(KCOM(I),I=1,2),NSEC,JBPM
87    WRITE(1,70)
70    FORMAT('?')
      READ(6,71,END=78)(LL(I),I=1,72)
71    FORMAT(72A1)
C
C     ANALISIS DE COMANDOS A TABLA DE MACROS
C
      CALL COMAC(LL,NB)
      IF(NB.EQ.1)GO TO 87
      DO 73 IN=2,21
      READ(5'IN)(KCOM(I),I=1,2),NSEC,JCO,M,K
      IF(LL(1).EQ.KCOM(1).AND.LL(2).EQ.KCOM(2))GO TO 74
73    CONTINUE
      GO TO 76
C
C     EJECUCION DE COMANDOS
C
74    CALL SIM(LL,IN,NSEC,JCO,M,K)
      GO TO 87
76    DO 77 I=1,5
77    L5(I)=LL(I)
C
C     FORMACION DEL NOMBRE DE UN MACRO
C
      CALL MAC(L5,NE,I1)
      IF(NE.EQ.0) GO TO 11
      IM1=I1-1
C
C     BUSCA SI YA EXISTE EL MACRO EN LA TABLA
C
      CALL RECOM(IM1,L5,N1,K)
      IF(N1.EQ.1)GO TO 15
C
```

```
      CALL SIMAC(LL,NEM,NPOC,NPUN)
      IF(NEM.EQ.0)GO TO 79
      GO TO 80
15    IM2=IM1+1
      DO 40 IRB=IM2,70
      IF(LL(IRB).NE.BLANCO)GO TO 82
40    CONTINUE
C
C     EJECUCION DEL MACROCOMANDO
C
      CALL EJEMA(K,MER)
      IF(MER.NE.0)GO TO 80
      GO TO 87
C
C     ESCRITURA DEL MACROCOMANDO EN LA TABLA
C
79    CALL ESMA(LL,NPUN,NPOC,MLL)
      IF(MLL.EQ.1)GO TO 81
      GO TO 87
11    WRITE(1,12)
12    FORMAT('*01*ERROR DE SINTAXIS EN MACROCOMANDO')
      GO TO 87
82    WRITE(1,16)
16    FORMAT('*04*ERROR:SE QUIERE CREAR UN MACRO EXISTENTE')
      GO TO 87
80    WRITE(1,13)
13    FORMAT('*02*ERROR EN DEFINICION DE MACRO')
      GO TO 87
81    WRITE(1,14)
14    FORMAT('*03*ERROR,LIBRERIA DE MACRO LLENA')
      GO TO 87
78    CALL EXIT
      END
CART ID 4EE2   DB ADDR  51F0   DB CNT   00C4


00/00/70  0258 HRS
PAGE 001


==>
```

```
T INT4
INT4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       CREACION DEL ARCHIVO DE INSTRUCCIONES
        DEFINE FILE2 (256,10,U,J2)
        DIMENSION INT(4)
        DATA INT/'0','0','0','0'/
        WRITE(1,10)
10      FORMAT('J',/,'XXX')
        READ(6,11)J
11      FORMAT(I3)
6       WRITE(1,7)
7       FORMAT('INT',1X,'NPRO',1X,'NDIR',1X,'NREG',/,'MMPPDRTT')
        READ(6,3,END=5)(INT(I),I=3,4),NPRO,NDIR,NREG,NTIME
3       FORMAT(2A1,I2,I1,I1,I2)
        M=4
        CALL HEDE4(INT,MOD,M)
        WRITE(1,8)(INT(I),I=1,4),MOD
8       FORMAT(4A1,2X,I6)
        WRITE(4'J)MOD,NPRO,NDIR,NREG,NTIME
        J=J+1
        GO TO 6
5       CALL EXIT
        END
CART ID 4EE2    DB ADDR   4DB0    DB CNT    0030


00/00/70   0259 HRS
PAGE 001


==>
```

```
T LECOD
LECOD.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       LECTURA DEL ARCHIVO DE INSTRUCCIONES

        DEFINE FILE 2 (256,10,U,J2)
        DIMENSION NNHEX(2),NBCD(2)
6       WRITE(1,10)
10      FORMAT('J',1X,'N',/,'XXXYYY')
        READ(6,11,END=5)J,N
11      FORMAT(2I3)
        IF(J.GT.198.OR.N.GT.198)GO TO 7
        WRITE(1,8)
8       FORMAT(5X,'ARCHIVO DE INSTRUCCIONES',/,'POSIC',3X,'OP',2X
       1,'CODIGO',1X,'DIREC',1X,'REGIS',1X,'TIEMPO')
        DO 12 K=J,N
        READ(2'K)NOD,NPRO,NDIR,NREG,NTIME
        CALL DEC4 (NOD,NNHEX,NBCD)
        WRITE(1,9)K,(NNHEX(I),I=1,2),NPRO,NDIR,NREG,NTIME
9       FORMAT(I6,2X,2A1,2X,5I6)
12      CONTINUE
        GO TO 6
7       WRITE(1,15)
15      FORMAT(2X,'ERROR:SE SOBREPASA EL ARCHIVO')
5       CALL EXIT
        END
CART ID 4EE2   DB ADDR  4E60   DB CNT   0034


00/00/70  0259 HRS
PAGE 001


==>
```

```
T COMAN
COMAN.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C      CREACION DEL ARCHIVO DE COMANDOS

       DEFINE FILE5(22,12,U,J5)
       DIMENSION KCOM(2)
       WRITE(1,10)
10     FORMAT('J',/,'XX')
       READ(6,11)J
11     FORMAT(I2)
6      WRITE(1,7)
7      FORMAT('KCOM(1)',1X,'KCOM(2)',1X,'NSEC',1X,'JESP',1X,'M',1X,'K',
      1/,'CCNNJJJJJJJJJMMKK')
       READ(6,3,END=5)(KCOM(I),I=1,2),NSEC,JESP,M,K
3      FORMAT(2A1,1I2,I9,2I2)
       WRITE(5'J)(KCOM(I),I=1,2),NSEC,JESP,M,K
       J=J+1
       GO TO 6
5      CALL EXIT
       END
CART ID 4EE2   DB ADDR  4DF0   DB CNT   002A


00/00/70  0300 HRS
PAGE 001


==>
```

```
T LCOM
LCOM.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       LECTURA DEL ARCHIVO DE COMANDOS

        DEFINE FILE 5(22,12,U,J5)
        DIMENSION LCOM(2)
6       WRITE(1,10)
10      FORMAT('J','N',/,'XXYY')
        READ(6,11,END=5)J,N
11      FORMAT(2I2)
        IF(J.GT.22.OR.N.GT.22)GO TO 15
        WRITE(1,21)
21      FORMAT(5X,'ARCHIVO DE COMANDOS',/,'POS',1X,'COM',
       16X,'CODIGO',3X,'ESPECIF',5X,'PROP1',5X,'PROP2')
        DO 22 L=J,N
        READ(5'L)(LCOM(I),I=1,2),LSEC,LESP,M,K
        WRITE(1,19)L,(LCOM(I),I=1,2),LSEC,LESP,M,K
19      FORMAT(I3,2X,2A1,2X,5I10)
22      CONTINUE
        GO TO 6
15      WRITE(1,16)
16      FORMAT('ERROR:SE SOBREPASA EL ARCHIVO')
5       CALL EXIT
        END
CART ID 4EE2    DB ADDR  4EA0    DB CNT    0032


00/00/70  0300 HRS
PAGE 001


==>
```

```
T CEROR
CEROR.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C
C       GRAVACION DE ERRORES EN EL ARCHIVO ERR
C
        DEFINE FILE 7(50,160,U,J7)
        DIMENSION LER(80)
        WRITE(1,5)
5       FORMAT('NN')
        READ(6,6)NN
6       FORMAT(I2)
10      IF(NN.GE.15)GO TO 7
        WRITE(1,8)
8       FORMAT('INI,IFI',/,'NNFF')
        READ(6,9,END=14)INI,IFI
9       FORMAT(2I2)
        WRITE(7'NN)INI,IFI
        NN=NN+1
        GO TO 10
7       WRITE (1,11)NN
11      FORMAT('ESCRIBA ERROR EN EL REGISTRO ',I2)
        READ(6,12,END=14)(LER(I),I=1,80)
12      FORMAT(80A1)
        WRITE(7'NN)(LER(I),I=1,80)
        NN=NN+1
        GO TO 7
14      CALL EXIT
        END
CART ID 4EE2   DB ADDR  4E20   DB CNT   0038


00/00/70  0300 HRS
PAGE 001


==>
```

```
T LEROS
LEROS.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       LECTURA DEL ARCHIVO DE ERRORES

        DEFINE FILE 7 (50,160,U,J7)
6       WRITE(1,10)
10      FORMAT('J',1X,'N',/,'XXYY')
        READ(6,11,END=5)J,N
11      FORMAT(2I2)
        IF(J.GT.13.OR.N.GT.13)GO TO 15
        WRITE(1,20)
20      FORMAT(5X,'ARCHIVO DE ERRORES',//)
        DO 21 K=J,N
        CALL LERR(K)
        WRITE(1,22)
22      FORMAT(5X,//)
21      CONTINUE
        GO TO 6
15      WRITE (1,16)
16      FORMAT(2X,'ERROR:SE SOBREPASA EL ARCHIVO')
5       CALL EXIT
        END
CART ID 4EE2   DB ADDR  4EE0   DB CNT   002E


00/00/70  0301 HRS
PAGE 001


==>
```

```
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       SUBRUTINA QUE REALIZA COMANDOS A MACROS

        SUBROUTINE COMAC (LL,NR)
        DIMENSION LL(72),LS(72),NCOM(20),NAM(5)
        INTEGER COMA/','/,BLANCO/' '/
        DATA NM/'M'/,NL/'L'/,NB/'B'/
        ICO=0
        NR=0
        IF(LL(1).EQ.NL.AND.LL(2).EQ.NM)GO TO 10
        IF(LL(1).EQ.NB.AND.LL(2).EQ.NM)GO TO 20
        GO TO 80
C       COMANDO PARA LISTAR MACROS :LM
10      NR=1
        READ(6'1)LIM
        DO11 I=2,LIM
        READ(6'I)(LS(LA),LA=1,60)
        WRITE(1,12)(LS(J),J=1,60)
12      FORMAT(60A1)
11      CONTINUE
        GO TO 80
C       COMANDOS PARA BORRAR MACROS :BM
20      NR=1
        DO 31 II=1,20
31      NCOM(II)=0
        DO 21 J1=1,72
        NBL=J1
        IF(LL(J1).EQ.BLANCO)GO TO 32
        IF(LL(J1).NE.COMA)GO TO 21
        ICO=ICO+1
        NCOM(ICO)=J1
21      CONTINUE
32      MI=NCOM(1)+1
        DO 22    I2=2,10
        IF(NBL.GT.MI)N=NBL-1
        IF(NCOM(I2).NE.0)N=NCOM(I2)-1
        NDIF=N-MI
        IF(NDIF.GT.5)GO TO 79
        MN=0
        DO 23 K1=MI,N
        MN=MN+1
23      NAM(MN)=LL(K1)
        MN=MN+1
        NAM(MN)=BLANCO
        CALL MAC(NAM,NE,I1)
        IF(NE.EQ.0)GO TO 79
        IM1=I1-1
        CALL RECOM(IM1,NAM,N1,K)
        IF(N1.EQ.0)GO TO 78
        READ(6'1)LIM
        LIM=LIM-1
        WRITE(6'1)LIM
        DO 26 J2=K,LIM
        JJ=J2+1
        READ (6'JJ)(LS(L1),L1=1,72)
 26     WRITE(6'J2)(LS(L1),L1=1,72)
        IF(NCOM(I2).EQ.0)GO TO 80
22      MI=NCOM(I2)+1
        GO TO 80
78      WRITE(1,77)
77      FORMAT('*05*ERROR NO EXITE EL MACRO QUE SE PIDE BORRAR')
        GO TO 80
79      WRITE(1,76)
```

```
T SIM
SIM.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       SUBRUTINA QUE REALIZA CADA COMANDO

        SUBROUTINE SIM(LL,IM,NSEC,JCO,M,K)
        DIMENSION LL(72),NOO(18),MUL(4),KCOM(2)
        INTEGER BLANCO/' '/,COMA/','/
        READ(5'19)(KCOM(I),I=1,2),NAX,JAX
        IF(JAX.EQ.0)GO TO 31
        WRITE(1,32)(LL(I),I=1,72)
32      FORMAT(/,72A1,/)
31      NDID=0
        M4=0
        DO 89 I=1,18
89      NOO(I)=0
        NSED=NSEC/10
        NSEU=NSEC-NSED*10
        NERR=0
        GO TO (10,20),NSED
10      GO TO (11,11,13,14,15,16,17,18,19),NSEU
20      GO TO (21,22,23,24,25,26,27,28),NSEU
C       COMANDOS PARA BASE DE SALIDA Y BASE DE ENTRADA
11      IF(LL(3).EQ.BLANCO.OR.LL(5).NE.BLANCO)GO TO 76
        CALL ESPE(LL(3),LL(4),JESP)
        CALL DEK(JESP,M,K)
        WRITE(5'IM)LL(1),LL(2),NSEC,JESP,M,K
        GO TO 90
C       COMANDO PARA SELECCIONAR REGISTROS
13      IF(LL(3).NE.BLANCO)GO TO 76
        JESP=0
        DO131 I=4,12
        NCAR=LL(I)
        CALL RENUM(NCAR,NUM)
131     JESP=JESP*10+NUM
        WRITE(5'IM)LL(1),LL(2),NSEC,JESP
        GO TO 90
C       COMANDO PARA VER REGISTROS
14      CALL DEBAS(K)
        GO TO 90
C       COMANDO PARA VER LA ULTIMA INSTRUCCION
15      READ(3'1)MPD,MOD,MDI,IEA,NXD,NAD,NBD,MCCRD,MSD
        CALL REAO(MDI,MOD)
        MM=4
        CALL DECE4(MOD,MUL,MM)
        WRITE(1,151)MUL
151     FORMAT(/,'ULT-INST=',4A1)
        GO TO 90
C       COMANDO PARA VER MEMORIA
16      IF(LL(3).NE.BLANCO.OR.LL(4).EQ.BLANCO)GO TO 76
        READ(5'3)KCOM1,KCOM2,NSEC,JBAS,M,K
        N=M/2
        NRD=1
        CALL DENT(NDID,NOO,M,NERR,LL,N,JBAS,NRD,M4)
        IF(NERR.EQ.1)GO TO 76
        NDD=NOO(1)
        CALL IMPM(NDID,NDD)
        GO TO 90
C       COMANDO PARA ESCOGER CODIGO
17      CALL ESPE(LL(4),LL(5),JESP)
        WRITE(5'IM)LL(1),LL(2),NSEC,JESP
        GO TO 90
```

```
      IF(LL(3).NE.BLANCO)GO TO 76
      CALL ESPE(LL(4),LL(5),JDE)
      WRITE(5'9)LL(1),LL(2),NSEC,JDE
      GO TO 90
C     COMANDO PARA TRANSFERIR CODIGO OBJETO
19    CALL GRAT
      GO TO 90
C     COMANDO PARA EJECUTAR O RASTREAR UN PROGRAMA
21    IF(LL(3).NE.BLANCO)GO TO 76
      READ(5'3)NAU1,NAU2,NSEC,JESP,M,K
      NRD=1
      N=M
      CALL DENT(NDID,NOO,M,NERR,LL,N,JESP,NRD,M4)
      NREP=NOO(1)
      IF(NERR.EQ.1)GO TO 76
      CALL EJET(NDID,NREP,JCO)
      GO TO 90
C     COMANDO PARA PONER VALORES EN MEMORIA
22    READ(5'3)KCOM1,KCOM2,NSEC,JESP,M,K
      N=M/2
      NRD=JESP+2
      CALL DENT(NDID,NOO,M,NERR,LL,N,JESP,NRD,M4)
222   IF(NERR.EQ.1)GO TO 76
      L=1
      NIR=NDID+NRD
      DO221 KI=NDID,NIR
      CALL WRITO(KI,NOO(L))
221   L=L+1
      JBPM=1
      READ(5'13)(KCOM(I),I=1,2),NSEC
      WRITE(5'13)(KCOM(I),I=1,2),NSEC,JBPM
      GO TO 90
C     COMANDO PARA CONTINUAR PONIENDO VALORES EN MEMORIA
23    READ(5'3)NAU1,NAU2,NSEC,JESP,M,K
      N=M/2
      NRD=JESP+2
      IF(LL(3).NE.BLANCO.OR.LL(4).EQ.BLANCO)GO TO 76
      READ(5'13)(KCOM(I),I=1,2),NSEC,JBPM
      IF(JBPM.EQ.0)GO TO 76
      M4=3
      CALL CONT
      NDID=NIR+1
      GO TO 222
C     COMANDO PARA PONER VALORES EN REGISTROS SIMULADOS
24    CALL ENRE(LL,NERR)
      IF(NERR.EQ.1)GO TO 76
      GO TO 90
C     COMANDO PARA LISTAR COMANDOS INGRESADOS
25    JESP=1
      WRITE(5'19)LL(1),LL(2),NSEC,JESP
      GO TO 90
C     COMANDO PARA NO LISTAR COMANDOS INGRESADOS
26    READ(5'19)LL(1),LL(2),NSEC,JESP
      JESP=0
      WRITE(5'19)LL(1),LL(2),NSEC,JESP
      GO TO 90
C     COMANDO PARA OBTENER EL MENSAJE DE UN ERROR
27    IF(LL(3).NE.BLANCO)GO TO 76
      CALL ESPE(LL(4),LL(5),NER)
      IF(NER.GT.13)GO TO 76
      CALL LERR(NER)
      GO TO 90
C     COMANDO PARA OBTENER LA DIRECCION DE LA ULTIMA
C     INSTRUCCION EJECUTADA
28    CALL ULTID
      GO TO 90
```

```
T MAC
MAC.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       SUBRUTINA QUE ANALIZA EL NOMBRE DEL MACRO

        SUBROUTINE MAC(L5,NE,I1)
        DIMENSION M(4,5),L5(5),LE(5)
        M(1,1)=2
        M(1,2)=3
        M(1,3)=3
        M(2,1)=2
        M(2,2)=2
        M(2,3)=4
        DO 5 J=1,3
        M(3,J)=3
5       M(4,J)=4
        DO 6 I=1,3
6       M(I,5)=0
        M(4,5)=1
        M(1,4)=3
        M(2,4)=3
        DO 7 K=1,5
7       LE(K)=L5(K)
        I1=1
        II=1
9       CALL EXPLO(LE(I1),NT)
        II=M(II,NT)
        NE=M(II,5)
        IF(NE.EQ.1.OR.I1.EQ.5)GO TO 8
        I1=I1+1
        GO TO 9
8       RETURN
        END
CART ID 4EE2   DB ADDR  4F30   DB CNT   0042


00/00/70  0303 HRS
PAGE 001


==>
```

```
T RECOM
RECOM.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       SUBRUTINA QUE VE EN LA LIBRERIA DE MACROS SI EXISTE EL MACRO

        SUBROUTINE RECOM(IM1,L5,N1,K)
        DIMENSION L5(5),NAM(5)
        INTEGER BLANCO /' '/
        N1=0
        READ(6'1)KDCOM
        DO 10 I=2,KDCOM
        READ(6'I)(NAM(II),II=1,5)
        DO 11 J=1,IM1
        IF(L5(J).NE.NAM(J))GO TO 10
11      CONTINUE
        IM2=IM1+1
        IF(NAM(IM2).NE.BLANCO)GO TO 10
        N1=1
        K=I
        GO TO 14
10      CONTINUE
14      RETURN
        END
CART ID 4EE2    DB ADDR  4560    DB CNT    002E

00/00/70  0304 HRS
PAGE 001

==>
```

```
T SIMAC
SIMAC.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       SUBRUTINA QUE COMPRUEBA LA SINTAXIS DEL MACRO

        SUBROUTINE SIMAC(LL,NEM,NPOC,NPUN)
        DIMENSION LL(72),LE(60),NPUN(10),NPOC(2),NBLC(10)
        INTEGER PUNTO/'.'/,CORE/'('/,COSE/')'/,BLANCO/' '/
        DO 6 I=1,10
        NBLC(I)=0
6       NPUN(I)=0
        NPOC(1)=0
        NPOC(2)=0
        I1=0
        I2=0
        LP=0
        I3=0
        NEM=1
        DO 9 I=1,60
9       LE(I)=LL(I)
        DO 10 I=1,60
        IF(LE(I).NE.CORE)GO TO 16
        I1=I1+1
        NPOC(1)=I
16      IF(LE(I).NE.PUNTO)GO TO 11
        LP=LP+1
        NPUN(LP)=I
11      IF(LE(I).NE.BLANCO)GO TO 12
        I3=I3+1
        NBLC(I3)=I
12      IF(LE(I).NE.COSE)GO TO 10
        I2=I2+1
        NPOC(2)=I
10      CONTINUE
        IF(NPOC(1).GT.NPOC(2).OR.I1.NE.1.OR.I2.NE.1)GO TO 15
        DO 14 K=1,10
        DO 14 L=1,10
        IF(NBLC(K)-NPUN(L).EQ.1)GO TO 15
14      CONTINUE
        NEM=0
15      RETURN
        END
CART ID 4EE2    DB ADDR   44F0    DB CNT    0054


00/00/70   0304 HRS
PAGE 001


==>
```

```
T EJEMA
EJEMA.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

      SUBROUTINE EJEMA(K,MER)
      DIMENSION LS(60),NPUN(10),NPOC(2),LR(72),KCOM(2)
      INTEGER BLANCO/' '/
      DO 19 IB=1,72
19    LR(IB)=BLANCO
      READ(6'K)(LS(I),I=1,60),(NPUN(J),J=1,10),(NPOC(L),L=1,2)
      I1=NPOC(1)+1
      M=0
      MER=1
      DO 10 IC=1,10
      I2=NPOC(2)-1
      IF(NPUN(IC).NE.0)I2=NPUN(IC)-1
      JR=0
      IF(I1.GT.I2)GO TO 16
      MER=0
      DO 12 M=I1,I2
      JR=JR+1
12    LR(JR)=LS(M)
      JR1=JR+1
      DO 17 N=JR1,72
17    LR(N)=BLANCO
      DO 13 IM=2,21
      READ(5'IM)(KCOM(I),I=1,2),NSEC,JCO,MN,KL
      IF(LR(1).EQ.KCOM(1).AND.LR(2).EQ.KCOM(2))GO TO 14
13    CONTINUE
      WRITE(1,15)
15    FORMAT('*08*ERROR EN COMANDO DEL MACRO')
      GO TO 11
14    CALL SIM(LR,IM,NSEC,JCO,MN,KL)
      IF(NPUN(IC).EQ.0)GO TO 11
      I1=NPUN(IC)+1
10    CONTINUE
      GO TO 11
16    MER=1
11    RETURN
      END
CART ID 4EE2   DB ADDR. 5160   DB CNT   004E


00/00/70  0305 HRS
PAGE 001


==>
```

```
T ESMA
ESMA.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

      SUBROUTINE ESMA (LL,NPUN,NPOC,MLL)
      DIMENSION LL(72),LE(60),NPUN(10),NPOC(2)
      MLL=1
      DO 10 I=1,60
10    LE(I)=LL(I)
      READ(6'1)LIM
      IF(LIM.GE.30)GO TO 20
      MLL=0
      LIM=LIM+1
      WRITE(6'1)LIM
      WRITE(6'LIM)(LE(K),K=1,60),(NPUN(J),J=1,10),(NPOC(M),M=1,2)
20    RETURN
      END
CART ID 4EE2   DB ADDR  4590   DB CNT   0020


00/00/70  0305 HRS
PAGE 001


==>
```

```
T HEDE4
HEDE4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C      SUBRUTINA QUE CAMBIA UN NUMERO EN BASE HEXADECIMAL
C      A BASE DECIMAL

            SUBROUTINE HEDE4(MOP,MOD,M)
            DIMENSION NAUX(4),NUMX(16),MOP(4)
            DATA NAUX/'0','0','0','0'/
            DATA NUMX/'0','1','2','3','4','5','6','7','8',
      1     '9','A','B','C','D','E','F'/
            IF(M.GT.4)GO TO 12
            DO5 I=1,M
            DO 6 J=1,16
            IF(MOP(I).NE.NUMX(J))GO TO 6
            NAUX(I)=J-1
6           CONTINUE
5           CONTINUE
            MOD=0
            DO 10 L=1,M
10          MOD=MOD*16+NAUX(L)
            RETURN
12          WRITE(1,13)
13          FORMAT('ERROR:NO SE CAMBIA DE HEXA. A DEC.')
            RETURN
            END
CART ID 4EE2   DB ADDR  4FC0   DB CNT   0032


00/00/70  1009 HRS
PAGE 001


==>
```

```
T DEC4
DEC4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
      SUBROUTINE DEC4(INDE,NHEX,NBCD)
C     SUBRUTINA QUE CAMBIA UN NUMERO DECIMAL A HEXADECIMAL
C     DE DOS DIGITOS

      DIMENSION NAUX(2),NHEX(2),NUMX(16),NBCD(2)
      DATA NUMX/'0','1','2','3','4','5','6','7','8','9',
     1'A','B','C','D','E','F'/
      NDIC=INDE
      DO5 I=1,2
      J=2-I
      M=16**J
      IF(NDIC.GE.M)GO TO 15
      NAUX(I)=0
      GO TO 5
15    NAUX(I)=NDIC/M
      NDIC=NDIC-NAUX(I)*M
5     CONTINUE
      DO 20 I=1,2
      DO 10 J=1,16
      K=J-1
      IF(NAUX(I).NE.K)GO TO 10
      NHEX(I)=NUMX(J)
      NBCD(I)=K
      GO TO 20
10    CONTINUE
20    CONTINUE
      RETURN
      END
CART ID 4EE2   DB ADDR  4460   DB CNT   003C


00/00/70  0307 HRS
PAGE 001


==>
```

```
T LERR
LERR.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       SUBRUTINA QUE LEE UN ERROR DEL ARCHIVO DE ERRORES

        SUBROUTINE LERR(NER)
        DIMENSION LER(80)
        NER1=NER+1
        READ(7'NER1)INI,IFI
        DO 5 I=INI,IFI
        READ(7'I)(LER(J),J=1,80)
        WRITE(1,6)(LER(J),J=1,80)
6       FORMAT(80A1)
5       CONTINUE
        RETURN
        END
CART ID 4EE2    DB ADDR   4B40    DB CNT    0020


00/00/70  0308 HRS
PAGE 001


==>
```

```
T ESPE
ESPE.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       SUBRUTINA QUE ESPECIFICA EL VALOR DE LA BASE

        SUBROUTINE ESPE(N1,N2,JES)
        NCAR1=N1
        CALL CARD(NCAR1,NUM1)
        NCAR2=N2
        CALL CARD(NCAR2,NUM2)
        JES=NUM1*10+NUM2
        RETURN
        END
CART ID 4EE2    DB ADDR   4390    DB CNT    001A


00/00/70  0308 HRS
PAGE 001


==>
```

```
T DEB
DEK.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       SUBRUTINA QUE ASIGNA PARAMETROS DE ACUERDO A LA BASE

        SUBROUTINE DEK(JES,M,K)
        IF(JES.EQ.2)GO TO 2
        IF(JES.EQ.8)GO TO 8
        IF(JES.EQ.16)GO TO 16
        IF(JES.EQ.10)GO TO 10
2       K=1
        M=16
        GO TO 11
8       K=2
        M=6
        GO TO 11
16      K=3
        M=4
        GO TO 11
10      K=4
        M=6
11      RETURN
        END
CART ID 4EE2   DB ADDR  4350   DB CNT   002E


00/00/70  0308 HRS
PAGE 001


==>
```

```
T RENUM
RENUM.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       SUBRUTINA QUE ASIGNA UN NUMERO A CADA REGISTRO

        SUBROUTINE RENUM(NC,NUM)
        DIMENSION NENT(9)
        DATA NENT/'P','O','I','E','X','A','B','C','S'/
        NUM=0
        DO5 I=1,9
        IF(NC.EQ.NENT(I))GO TO 6
5       CONTINUE
        GO TO 7
6       NUM=I
7       RETURN
        END
CART ID 4EE2    DB ADDR  4300    DB CNT    0020


00/00/70  0309 HRS
PAGE 001


==>
```

```
I DEBAS
DEBAS.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C      SUBRUTINA QUE IMPRIME LOS REGISTROS DE ACUERDO A LA BASE

       SUBROUTINE DEBAS(LL)
       DIMENSION NC16(9,16),NLEC(9),NENT(9)
       DATA NENT/'P','O','I','E','X','A','B','C','S'/
       READ(3'1)(NLEC(I),I=1,9)
       READ(5'2)NAU1,NAU2,NAU3,JBAS,M
       WRITE(1,26)JBAS
26     FORMAT('BASE=',I6)
       N=9
       CALL CO210(NLEC,JBAS,M,NC16,N)
       READ(5'4)N1,N2,N3,JESP
       NK=M
       DO 25 I=1,9
       M=NK
       J=9-I
       NCAR=JESP/10**J
       JESP=JESP-NCAR*10**J
       IF(NCAR.EQ.0)GO TO 25
       IF(NCAR.EQ.2.OR.NCAR.EQ.6.OR.NCAR.EQ.7.OR.NCAR.EQ.8)M=M/2
       NKM=NK-M+1
       IF(JBAS.EQ.16)GO TO 23
       WRITE(1,22)NENT(NCAR),(NC16(NCAR,J),J=NKM,NK)
22     FORMAT(5X,A2,'=',16I2)
       GO TO 25
23     WRITE(1,24)NENT(NCAR),(NC16(NCAR,J),J=NKM,NK)
24     FORMAT(5X,A2,'=',16A2)
25     CONTINUE
       RETURN
       END
CART ID 4EE2   DB ADDR  44A0   DB CNT   0042


00/00/70  0309 HRS
PAGE 001


==>
```

```
T READ
READ.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C       SUBRUTINA QUE LEE DE LA MEMORIA 0000

        SUBROUTINE READ(MPD,MOD)
        IF(MPD.NE.0)GO TO 5
        READ(1'65536)MOD
        GO TO 6
5       READ(1'MPD)MOD
6       RETURN
        END
CART ID 4EE2    DB ADDR  4290    DB CNT    0016


00/00/70  0309 HRS
PAGE 001


==>
```

```
T DECE4
DECE4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C       SUBRUTINA QUE CAMBIA UN NUMERO DECIMAL A HEXADECIMAL
C       DE 4 DIGITOS

        SUBROUTINE DECE4(INDE,NHEX,L)
        DIMENSION NAUX(4),NHEX(4),NUMX(16)
        DATA NUMX/'0','1','2','3','4','5','6','7','8',
       1'9','A','B','C','D','E','F'/
        DATA NAUX/'0','0','0','0'/
        IF(L.GT.4)GO TO 16
        NDIC=INDE
        DO 5 I=1,L
        J=L-I
        M=16**J
        IF(NDIC.GE.M)GO TO 15
        NAUX(I)=0
        GO TO 5
15      NAUX(I)=NDIC/M
        NDIC=NDIC-NAUX(I)*M
5       CONTINUE
        DO 10 I=1,L
        DO 10 J=1,16
        K=J-1
        IF(NAUX(I).NE.K)GO TO 10
        NHEX(I)=NUMX(J)
10      CONTINUE
        RETURN
16      WRITE(1,17)
17      FORMAT(2X,'ERROR:NO SE CAMBIA DE DEC. A HEXAD.')
        RETURN
        END
CART ID 4EE2    DB ADDR   4F80    DB CNT    0040


00/00/70   1010 HRS
PAGE 001


==>
```

```
T DENT
DENT.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       SUBRUTINA QUE ANALIZA DATOS DE ENTRADA ,DE ACUERDO
C       A UNA BASE DE ENTRADA Y PONIENDO EN DECIMAL

        SUBROUTINE DENT(NDID,NDD,M,NERR,LL,N,JBAS,NRD,M4)
        DIMENSION KD16(16),LL(72),KD8(16),NDD(18)
        INTEGER BLANCO /' '/,COMA /','/
        DO 10 I=1,M
        I3=I+3
        IF(LL(I3).EQ.BLANCO)GO TO 19
10      KD16(I)=LL(I3)
        M1=M
        CALL CB16(KD16,NDID,M1,JBAS)
        IF(NDID.GE.65536)GO TO 15
        M4=M+4
        NERR=0
        IF(LL(M4).EQ.BLANCO.OR.LL(M4).NE.COMA)GO TO 19
        ENTRY CONT
        DO40 I=1,NRD
        DO20 J=1,N
        JM=J+M4
        IF(LL(JM).EQ.BLANCO)GO TO 19
20      KD8(J)=LL(JM)
        N1=N
        CALL CB16(KD8,NDD,N1,JBAS)
        LIM2=256
        IF(N.EQ.M)LIM2=65536
        IF(NDD.GE.LIM2)GO TO 16
        NDD(I)=NDD
        M4=M4+N+1
        M5=M4+1
        IF(LL(M4).EQ.BLANCO.AND.LL(M5).EQ.BLANCO)GO TO 14
        IF(LL(M4).NE.COMA)GO TO 19
40      CONTINUE
        GO TO 13
19      NERR=1
        GO TO 13
14      NRD=I-1
        GO TO 13
15      WRITE(1,17)
17      FORMAT('*09*ERROR:DIRECCIONAMIENTO ILEGAL')
        GO TO 19
16      WRITE(1,18)
18      FORMAT('*10*ERROR:VALOR DE ENTRADA SOBREPASA LA CAPACIDAD')
        GO TO 19
13      RETURN
        END
CART ID 4EE2   DB ADDR  50F0   DB CNT   0062


00/00/70  0310 HRS
PAGE 001


==>
```

```
T IMPM
IMPM.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       SUBRUTINA QUE IMPRIME LAS DIRECCIONES Y CONTENIDOS DE MEMORIA

        SUBROUTINE IMPM(NDID,NDO)
        DIMENSION ND(9),MO(9),NC16(9,16),NK16(9,16)
        MN=NDID+NDO-1
        READ(5'2)KCOM1,KCOM2,NSEC,JBAS,M,K
        WRITE(1,66)
66      FORMAT(2X,'DIRECCION',10X,'CONTENIDO')
        DO84 JJ=NDID,MN
        CALL REAO(JJ,MO(1))
        N=1
        ND(1)=JJ
        CALL CO210(ND,JBAS,M,NC16,N)
        L=M/2
        CALL CO210(MO,JBAS,L,NK16,N)
        GO TO (2,8,16,8),K
2       WRITE(1,3)(NC16(1,J),J=1,M),(NK16(1,J),J=1,L)
3       FORMAT(2X,16I1,2X,8I2)
        GO TO 84
8       WRITE(1,9)(NC16(1,J),J=1,M),(NK16(1,J),J=1,L)
9       FORMAT(2X,6I1,12X,3I1)
        GO TO 84
16      WRITE(1,17)(NC16(1,J),J=1,4),(NK16(1,J),J=1,L)
17      FORMAT(2X,4A1,15X,2A1)
84      CONTINUE
        RETURN
        END
CART ID 4EE2    DB ADDR   43F0    DB CNT    003C


00/00/70  0311 HRS
PAGE 001


==>
```

```
*EXTENDED PRECISION

C       SUBRUTINA QUE TRANSFIERE DEL ARCHIVO ENSAMBLADO
C       AL ARCHIVO DE MEMORIA

        SUBROUTINE GRAT
        DIMENSION L(26),LD(4),LOP(4),LR1(4),LR2(4),M(70)
        INTEGER AST/'*'/,BLANCO/' '/
        DATA LD/'0','0','0','0'/,LOP/'0','0','0','0'/
        DATA LR1/'0','0','0','0'/,LR2/'0','0','0','0'/
        DATA L(1)-(10)/'A','B','C','D','E','F','G','H','I','J'/
        DATA L(11)-(20)/'K','L','M','N','O','P','Q','R','S','T'/
        DATA L(21)-(26)/'U','V','W','X','Y','Z'/
        NOP=0
        LOR1=0
        NBO=1
        MD=4
        LOR2=0
        N=2
        READ(4'1)N1
        N2=N1-1
        J=2
5       IF(J.GT.N1)GO TO 15
        READ(4'J)(M(I),I=1,70)
        IF(M(1).EQ.AST)GO TO 14
        IF(M(25).EQ.AST)GO TO 10
        IF(M(32).EQ.L(14).AND.M(33).EQ.L(1).AND.M(34).EQ.L(13))GO TO 10
        IF(M(32).EQ.L(15).AND.M(33).EQ.L(18).AND.M(34).EQ.L(7))GO TO 10
        IF(M(32).EQ.L(5).AND.M(33).EQ.L(17).AND.M(34).EQ.L(21))GO TO 10
        IF(M(32).EQ.L(18).AND.M(33).EQ.L(13).AND.M(34).EQ.L(2))GO TO 10
        IF(M(32).EQ.L(6).AND.M(33).EQ.L(3).AND.M(34).EQ.L(2))GO TO 11
        IF(M(32).EQ.L(6).AND.M(33).EQ.L(3).AND.M(34).EQ.L(3))GO TO 11
        IF(M(32).EQ.L(5).AND.M(33).EQ.L(14).AND.M(34).EQ.L(4))GO TO 15
        DO 20 K=1,4
        KM=K+11
20      LD(K)=M(KM)
        IF(NBO.NE.1)GO TO 11
        WRITE(1,17)N2,(LD(I),I=1,4)
17      FORMAT(/,2X,'UN PROGRAMA DE :',I3,'LINEAS ',/,
       1'DESDE LA LOCALIDAD :',4A1,'(HEXADECIMAL) FUE CARGADO')
        NBO=0
11      LD(1)=M(12)
        LD(2)=M(13)
        LD(3)=M(14)
        LD(4)=M(15)
        CALL HEDE4 (LD,NDID,MD)
        LOP(1)=M(17)
        LOP(2)=M(18)
        CALL HEDE4 (LOP,NOP,N)
        CALL WRITO(NDID,NOP)
        IF(M(20).EQ.BLANCO)GO TO 10
        LR1(1)=M(20)
        LR1(2)=M(21)
        CALL HEDE4 (LR1,LOR1,N)
        NDID=NDID+1
        CALL WRITO (NDID,LOR1)
        IF(M(22).EQ.BLANCO)GO TO 10
        LR2(1)=M(22)
        LR2(2)=M(23)
        CALL HEDE4 (LR2,LOR2,N)
        NDID=NDID+1
        CALL WRITO(NDID,LOR2)
10      J=J+1
        GO TO 5
14      WRITE(1,13)
```

```
T EJET
EJET.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C      SUBRUTINA QUE EJECUTA LAS INSTRUCCIONES

       SUBROUTINE EJET(NDID,NREP,JESP)
C      DIMENSIONAMIENTO DE ARREGLOS
       LOGICAL LA8(8),LB8(8),LM8(8),LR8(8),LC8(8),LL,AXL,NNOV,NC8(8)
       DIMENSION NA8(8),NB8(8),NCCR(8),NLEC(9),NC16(9,16),NHEX(2)
      1,NV8(8),NM8(8),NBCD(2),INT(198,5),NMON(4),MNO(108,16)
       INTEGER BLANCO/' '/
       DATA LR,LS,LI,LN,LO/'R','S','I','N','O'/
       READ(3'1)MPD,MOOD,MDI,IEA,NXD,NAD,NBD,MCCRD,MSD,NTIM
       READ(5'8)KAX1,KAX2,NSAX,NESP
       MPD=NDID
       IEA=NDID
       CALL READ(MPD,MOOD)
       MDI=NDID
       NTIM=0
       WRITE (3'1)MPD,MOOD,MDI,IEA,NXD,NAD,NBD,MCCRD,MSD,NTIM
       READ(5'9)KCOM1,KCOM2,NSEC,JDE
       IF(NESP.EQ.0)GO TO 9
       DO 8 I8=2,108
8      READ(8'I8)(MNO(I8,JM),JM=1,16)
9      JK5=0
       LBRAN=0
       LPRIN=0
       JB=0
       NUMBY=0
       DO 1 ITT=2,198
1      READ(2'ITT)(INT(ITT,JT),JT=1,5)
       MODO=MOOD
       IEAD=IEA
        JK=1
       NH=0
       NC=0
       NN=0
       NV=0
       NZ=0
       WRITE(1,3)
7      READ(3'1)MPD,MOOD,MDI,IEA,NXD,NAD,NBD,MCCRD,MSD,NTIM
       NLEC(1)=MDI
       NLEC(2)=MODO
       NLEC(3)=MPD
       NLEC(4)=IEAD
       NLEC(5)=NXD
       NLEC(6)=NAD
       NLEC(7)=NBD
       NLEC(8)=MCCRD
       NLEC(9)=MSD
       M4=4
       JBAS=16
       N9=9
       CALL COND4(MCCRD,NCCR)
       CALL CO210(NLEC,JBAS,M4,NC16,N9)
       IF(NESP.EQ.0)GO TO 85
       CALL NOMI(MODO,MNO,NMON)
       GO TO 86
85     NMON(1)=NC16(2,3)
       NMON(2)=NC16(2,4)
       NMON(3)=BLANCO
```

```
      JK5=JK5+1
      JK4=JK4+1
      GO TO (81,82,83,84,70),JESP
81    IF(JK.GT.NREP)GO TO 4
      GO TO 2
82    JK1=JK-1
      IF(JK1.GT.NREP)GO TO 87
      IF(JK5.GE.JDE)GO TO 4
      GO TO 70
83    IF(LBRAN.GT.NREP)GO TO 87
      IF(JB.EQ.1.AND.LPRIN.GE.JDE)GO TO 4
      IF(JB.EQ.1)GO TO 70
      GO TO 2
84    IF(JK4.GT.NREP)GO TO 4
      GO TO 2
4     WRITE(1,3)
3     FORMAT(/,3X,' I ',3X,' O ',3X,' P ',3X,' E ',3X,' X ',
     13X,' A ',3X,'B ',4X,'HINZVC',3X,'S',8X,'T')
      JK5=0
      LPRIN=0
70    WRITE(1,71)(NC16(1,J),J=1,4),(NMON(J),J=1,4),
     1((NC16(I,J),J=1,4),I=3,5),((NC16(I,J),J=3,4),I=6,7),
     1NCCR(6),NCCR(5),NCCR(4),NCCR(3),NCCR(2),NCCR(1),
     1(NC16(9,J),J=1,4),NTIM
71    FORMAT(3X,4A1,2X,4A1,2X,3(4A1,2X),2(2A1,4X),6I1,2X,4A1,1X,I8)
      JB=0
      GO TO (87,2,2,87,87),JESP
C     ENCONTRAR LA INSTRUCCION EN ARREGLO INT
C
2     CALL READ(MPD,MIN)
      DO 88 II=2,198
      IF(MIN.EQ.INT(II,1))GO TO 101
88    CONTINUE
      WRITE(1,89)
89    FORMAT(/,'*12*ERROR EN OPERANDO')
      GO TO 87
101   NPRO=INT(II,2)
      LDR=INT(II,3)
      NREG=INT(II,4)
      NT=INT(II,5)
      NTIM=NTIM+NT
      MODO=MOOD
      IF(LDR.EQ.1.AND.(NREG.EQ.3.OR.NREG.EQ.4))GO TO 102
      NUMBY=2
      GO TO 103
102   NUMBY=3
C     BUSQUEDA DEL OPERANDO EN BASE AL DIRECCIONAMIENTO
103   GO TO(90,91,92,93,94,95),LDR
90    IEA=IEA+1
      GO TO 97
91    IEA=IEA+1
      READ(1'IEA)LAX
      IEA=LAX
      NUMBY=2
      GO TO 97
92    IEA=IEA+1
      READ(1'IEA)LAX
      IEA=LAX+NXD
      NUMBY=2
      GO TO 97
93    IEA=IEA+1
      READ(1'IEA)LAX1
      IEA=IEA+1
      READ(1'IEA)LAX
```

```
      NUMBY=3
      GO TO 97
94    NUMBY=1
      GO TO 97
95    NUMBY=2
      IEA=IEA+1
97    CALL REDOS(IEA)
      CALL REAO(IEA,MOD)
      IEAD=IEA
      MDI=MPD
      MPD=MPD+NUMBY
C     DECODIFICACION DE INSTRUCCION
      MDD=NPRO/10
      MUD=NPRO-(MDD*10)
      GO TO(10,20,30,40,50,60),MDD
10    GO TO(11,12,13,14,15,16,17,18,19),MUD
20    GO TO(21,22,23,24,25,26,27,28,29),MUD
30    CALL COND4(NAD,NA8)
      CALL COND4(NBD,NB8)
      CALL COND4(MOD,NM8)
      GO TO(31,32,33,34,35,36,37,38),MUD
40    GO TO(41,42,43,44,45,46,47),MUD
50    CALL NUML4(MCCRD,NC8)
      AXL=.TRUE.
      LBRAN=LBRAN+1
      LPRIN=LPRIN+1
      JB=1
      GO TO(51,52,53,54,55,56,57,58,59),MUD
60    GO TO (61,62,63,64),MUD
11    NCX=0
      NBX=MOD
      GO TO(111,112,113),NREG
111   NAX=NAD
105   NRD=NAX+NBX+NCX
      MCD=NRD
      CALL ME254(NRD)
      IF(MUD.NE.6)NAD=NRD
      GO TO 115
112   NAX=NBD
110   NRD=NAX+NBX+NCX
      MCD=NRD
      CALL ME254(NRD)
      IF(MUD.NE.6)NBD=NRD
      GO TO 115
113   NBX=NBD
      NAX=NAD
      GO TO 105
115   IF(MUD.EQ.1.OR.MUD.EQ.2)GO TO 109
      GO TO 107
109   CALL HAC4(NAX,NBX,NCX,NH)
      CALL CR4(MCD,NC)
      GO TO 126
107   CALL BORR4(NAX,KBX,KCX,NC)
126   CALL NE4(NRD,NN)
      CALL CER4(NRD,NZ)
      CALL SOBR4(NAX,NBX,NCX,NRD,NV)
      NCCR(6)=NH
      NCCR(4)=NN
      NCCR(3)=NZ
      NCCR(2)=NV
      NCCR(1)=NC
5     CALL CONB4(NCCR,MCCRD)
6     IEA=MPD
      CALL REAO(MPD,MOOD)
      WRITE(3'1)MPD,MOOD,MDI,IEA,NXD,NAD,NBD,MCCRD,MSD,NTIM
```

```
      CALL ARRE4(NCCR,K1CRD)
      NCX=K1CRD
      GO TO (111,112),NREG
13    CALL  NUML4(NAD,LA8)
      CALL NUML4(NBD,LB8)
      CALL NUML4(MOD,LM8)
      GO TO(131,132),NREG
131   DO134 I=1,8
134   LR8(I)=LA8(I).AND.LM8(I)
      CALL LOGN4(LR8,NRD)
      IF(MUD.EQ.3)NAD=NRD
      GO TO 133
132   DO135 I=1,8
135   LR8(I)=LB8(I).AND.LM8(I)
      CALL LOGN4(LR8,NRD)
      IF(MUD.EQ.3)NBD=NRD
133   CALL NE4(NRD,NN)
      CALL CER4(NRD,NZ)
      NCCR(4)=NN
      NCCR(3)=NZ
      NCCR(2)=0
      GO TO 5
14    GO TO 13
15    GO TO(151,152,153),NREG
153   MM=0
      CALL WRITO(IEA,MM)
      GO TO 67
151   NAD=0
      GO TO 67
152   NBD=0
67    NCCR(1)=0
      NCCR(2)=0
      NCCR(3)=1
      NCCR(4)=0
      GO TO 5
16    NCX=0
      KCX=0
      GO TO(161,162,163),NREG
161   NAX=NAD
      NBX=MOD
      KBX=MOD
      NBX=256-NBX
      GO TO 105
162   NAX=NBD
      NBX=MOD
      KBX=MOD
      NBX=256-NBX
      GO TO 110
163   NAX=NAD
      NBX=NBD
      KBX=NBD
      NBX=256-NBD
      GO TO 105
28    GO TO 16
29    CALL ARRE4(NCCR,KCCRD)
      KCX=KCCRD
      NCX=256-KCX
      GO TO(161,162),NREG
17    NX=255
178   GO TO(171,172,173),NREG
171   IF(NAD.LT.128)GO TO 5
      NAD=NX-NAD
      NRD=NAD
      CALL ME254(NAD)
```

```
      NBD=NX-NPD
      NRD=NBD
      CALL ME254(NRD)
      GO TO 177
173   IF(MOD.LT.128)GO TO 5
      MEM=NX-MOD
      NRD=MEM
      CALL ME254(MEM)
      CALL WRITO(IEA,MEM)
177   IF(MUD.EQ.7)GO TO 175
      CALL SOB4(NRD,NV)
      CALL CAR4(NRD,NC)
      GO TO 174
175   NCCR(2)=0
      NCCR(1)=1
174   CALL NE4(NRD,NN)
      CALL CER4(NRD,NZ)
      NCCR(4)=NN
      NCCR(3)=NZ
      GO TO 5
18    NX=256
      GO TO 178
19    CALL DEC4(NAD,NHEX,NBCD)
      NAX=NAD
      NBX=0
      IF(NBCD(2).GT.9.OR.NCCR(6).EQ.1)GO TO 192
      GO TO 193
192   NC=0
      NBX=NBX+06
193   IF(NBCD(1).GT.9.OR.NCCR(1).EQ.1)GO TO 194
      GO TO 195
194   NBX=NBX+96
      NC=1
      GO TO 196
195   IF(NBCD(2).GT.9.AND.NBCD(1).EQ.9)GO TO 194
196   NRD=NAD+NBX
      IF(NRD.GT.256)NRD=NRD-256
      NAD=NRD
      GO TO 126
21    NN1=255
      NN2=128
210   GO TO(211,212,213),NREG
211   NVD=NAD
      NAD=NAD+NN1
      CALL ME254(NAD)
      NRD=NAD
      GO TO 217
212   NVD=NBD
      NBD=NBD+NN1
      CALL ME254(NBD)
      NRD=NBD
      GO TO 217
213   NVD=MOD
      MOD=MOD+NN1
      CALL ME254(MOD)
      NRD=MOD
      CALL WRITO(IEA,MOD)
217   CALL NE4(NRD,NN)
      CALL CER4(NRD,NZ)
      IF(NVD.EQ.NN2)GO TO 218
      NV=0
      GO TO 219
218   NV=1
219   NCCR(4)=NN
```

```
      GO TO 5
 22   CALL NUML4(NAD,LA8)
      CALL NUML4(NBD,LB8)
      CALL NUML4(MOD,LM8)
      GO TO (221,222),NREG
221   DO224 I=1,8
224   LR8(I)=.NOT.LA8(I).AND.LM8(I).OR.LA8(I).AND.(.NOT.LM8(I))
      CALL LOGN4(LR8,NAD)
      NRD=NAD
      GO TO 133
222   DO225 I=1,8
225   LR8(I)=.NOT.LB8(I).AND.LM8(I).OR.LB8(I).AND.(.NOT.LM8(I))
      CALL LOGN4(LR8,NBD)
      NRD=NBD
      GO TO 133
23    NN1=1
      NN2=127
      GO TO 210
24    GO TO (241,242)NREG
241   NAD=MOD
      NRD=NAD
      GO TO 243
242   NBD=MOD
      NRD=NBD
243   GO TO 133
25    CALL NUML4(MOD,LM8)
      GO TO(251,252),NREG
251   CALL NUML4(NAD,LA8)
      DO253 I=1,8
253   LR8(I)=LA8(I).OR.LM8(I)
      CALL LOGN4(LR8,NRD)
      NAD=NRD
      GO TO 133
252   CALL NUML4(NBD,LB8)
      DO254 I=1,8
254   LR8(I)=LB8(I).OR.LM8(I)
      CALL LOGN4(LR8,NBD)
      NRD=NBD
      GO TO 133
26    GO TO (261,262),NREG
261   CALL WRITO(MSD,NAD)
      GO TO 263
262   CALL WRITO(MSD,NBD)
263   MSD=MSD-1
      GO TO 6
27    MSD=MSD+1
      GO TO (271,272),NREG
271   CALL REAO(MSD,NAD)
      GO TO 273
272   CALL REAO(MSD,NBD)
273   GO TO 6
31    NAX=NCCR(1)
310   GO TO (311,312,313),NREG
311   NCCR(1)=NA8(8)
      NAD=NAD*2+NAX
      CALL ME254(NAD)
      NRD=NAD
      GO TO 314
312   NCCR(1)=NB8(8)
      NBD=NBD*2+NAX
      CALL ME254(NBD)
      NRD=NBD
      GO TO 314
313   NCCR(1)=NM8(8)
```

```
      NRD=MOD
      CALL WRITO(IEA,MOD)
314   CALL NE4(NRD,NN)
      AXL=.TRUE.
      NCCR(4)=NN
      CALL CER4 (NRD,NZ)
      NCCR(3)=NZ
      CALL CONB4(NCCR,MCCRD)
      CALL NUML4(MCCRD,LC8)
      LL=.NOT.LC8(4).AND.LC8(1).OR.LC8(4).AND.(.NOT.LC8(1))
      IF(LL.AND.AXL)GO TO 315
      NCCR(2)=0
      GO TO 316
315   NCCR(2)=1
316   GO TO 5
32    NCX=NCCR(1)
320   GO TO (321,322,323),NREG
321   CALL RO4(NA8,NCCR,NAD,NCX)
      NRD=NAD
      GO TO 314
322   CALL RO4(NB8,NCCR,NBD,NCX)
      NRD=NBD
      GO TO 314
323   CALL RO4(NM8,NCCR,MOD,NCX)
      CALL WRITO(IEA,MOD)
      NRD=MOD
      GO TO 314
33    NAX=0
      GO TO 310
34    GO TO (341,342,343),NREG
341   NCX=NA8(8)
      GO TO 321
342   NCX=NB8(8)
      GO TO 322
343   NCX=NM8(8)
      GO TO 323
35    NCX=0
      GO TO 320
36    GO TO (361,362),NREG
361   CALL WRITO(IEA,NAD)
      NRD=NAD
      GO TO 366
362   CALL WRITO(IEA,NBD)
      NRD=NBD
366   CALL NE4(NRD,NN)
      CALL CER4(NRD,NZ)
      NCCR(4)=NN
      NCCR(3)=NZ
      NCCR(2)=0
      GO TO 5
37    GO TO (371,372),NREG
371   NBD=NAD
      NRD=NBD
      GO TO 366
372   NAD=NBD
      NRD=NAD
      GO TO 366
38    GO TO (381,382,383),NREG
383   NRD=MOD
      GO TO 384
381   NRD=NAD
      GO TO 384
382   NRD=NBD
384   NCCR(1)=0
```

```
      CALL REDOS(IEA)
      CALL READ(IEA,MOD1)
      NMOD=MOD*256+MOD1
      NCOM=2**16
      NMOD=NCOM-NMOD
      NRD=NXD+NMOD
      CALL REDOS(NRD)
      CALL CER4(NRD,NZ)
      NOV=(NXD/256)+(256-MOD)
      CALL ME254(NOV)
      CALL COND4(NOV,NV8)
      IF(NV8(8).EQ.1)GO TO 412
      NCCR(4)=0
      GO TO 413
412   NCCR(4)=1
413   NC2=256-MOD1
      NXX=NXD
      CALL ME254(NXX)
      NRR=NXX+NC2
      CALL ME254(NRR)
      NCER=0
      CALL SOBR4(NXX,NC2,NCER,NRR,NV)
      NCCR(3) =NZ
      NCCR(2) =NV
      GO TO 5
42    MOS=2**16-1
420   GO TO (421,422),NREG
421   NXD=NXD+MOS
      CALL REDOS(NXD)
      CALL CER4(NXD,NZ)
      NCCR(3)=NZ
      GO TO 423
422   MSD=MSD+MOS
      CALL REDOS(MSD)
423   GO TO 5
43    MOS=1
      GO TO 420
44    IEA=IEA+1
      CALL REDOS(IEA)
      CALL READ(IEA,MOD1)
      GO TO (441,441,441,442),NREG
441   NXD=MOD*256+MOD1
      NRD=NXD
      GO TO 443
442   MSD=MOD*256+MOD1
      NRD=MSD
443   NRR=NRD/256
444    CALL BALD4(NRR,NCCR,NRD)
      GO TO 5
45    GO TO (451,452),NREG
451   NRD=NXD
453   NRR=NRD/256
      NRH=NRR
      NRL=NRD-NRR*256
      CALL WRITO(IEA,NRH)
      IEA=IEA+1
      CALL REDOS(IEA)
      CALL WRITO(IEA,NRL)
      GO TO 444
452   NRD=MSD
      GO TO 453
46    MOS=2**16-1
      MSD=NXD+MOS
      CALL REDOS(MSD)
```

```
      CALL REDOS(NXD)
      GO TO 6
51    NNOV=.NOT.NC8(4).AND.NC8(2).OR.NC8(4).AND.(.NOT.NC8(2))
      GO TO (511,512,513,514,515,516,517,518,519),NREG
511   IF(MOD.GT.127)GO TO 471
      MPD=MPD+MOD
      CALL REDOS(MPD)
      GO TO 6
471   MODC=256-MOD
      NCOM=2**16
      MODCC=NCOM-MODC
      MPD=MPD+MODCC
      CALL REDOS(MPD)
      GO TO 6
512   IF(NCCR(1).EQ.0)GO TO 511
      GO TO 6
513   IF(NCCR(1).EQ.1)GO TO 511
      GO TO 6
514   IF(NCCR(3).EQ.1)GO TO 511
      GO TO 6
515   IF(NNOV.AND.AXL)GO TO 6
      GO TO 511
516   IF(NC8(3).OR.NNOV)GO TO 6
      GO TO 511
517   IF(NC8(1).OR.NC8(3))GO TO 6
      GO TO 511
518   IF(NC8(3).OR.NNOV)GO TO 511
      GO TO 6
519   IF(NC8(1).OR.NC8(3))GO TO 511
      GO TO 6
52    NNOV=.NOT.NC8(4).AND.NC8(2).OR.NC8(4).AND.(.NOT.NC8(2))
      GO TO(521,522,523,524,525,526),NREG
521   IF(NNOV.AND.AXL)GO TO 511
      GO TO 6
522   IF(NCCR(4).EQ.1)GO TO 511
      GO TO 6
523   IF(NCCR(3).EQ.0)GO TO 511
      GO TO 6
524   IF(NCCR(2).EQ.0)GO TO 511
      GO TO 6
525   IF(NCCR(2).EQ.1)GO TO 511
      GO TO 6
526   IF(NCCR(4).EQ.0)GO TO 511
      GO TO 6
53    MPDSH=MPD/256
      MPDSL=MPD-MPDSH*256
      CALL WRITO(MSD,MPDSL)
      CALL REMOS(MSD)
      CALL WRITO(MSD,MPDSH)
      CALL REMOS(MSD)
      GO TO 511
54    MPD=IEA
      GO TO 6
55    MPD=IEA
      GO TO 6
56    MPDSH=MPD/256
      MPDSL=MPD-MPDSH*256
      CALL WRITO(MSD,MPDSL)
      CALL REMOS(MSD)
      CALL WRITO(MSD,MPDSH)
      CALL REMOS(MSD)
      MPD=IEA
      GO TO 6
57    GO TO 6
```

```
      CALL REAO(MSD,MCCRD)
      MSD=MSD+1
      CALL REDOS(MSD)
      CALL REAO(MSD,NBD)
      MSD=MSD+1
      CALL REDOS(MSD)
      CALL REAO(MSD,NAD)
      MSD=MSD+1
      CALL REDOS(MSD)
      CALL REAO(MSD,MAXH)
      MSD=MSD+1
      CALL REDOS(MSD)
      CALL REAO(MSD,MAXL)
      MSD=MSD+1
      CALL REDOS(MSD)
      CALL REAO(MSD,MPCH)
      MSD=MSD+1
      CALL REDOS(MSD)
      CALL REAO(MSD,MPCL)
      NXD=MAXH*256+MAXL
      MPD=MPCH*256+MPCL
      GO TO 6
59    MSD=MSD+1
      CALL REDOS(MSD)
      CALL REAO(MSD,MPCH)
      MSD=MSD+1
      CALL REDOS(MSD)
      CALL REAO(MSD,MPCL)
      MPD=MPCH*256+MPCL
      GO TO 6
61    CALL GUARD(NXD,MPD,MSD,IEA,NAD,NBD,MCCRD)
      GO TO (611,612),NREG
611   JESP=5
      GO TO 6
613   READ(1'MEM1)MPCH
      READ(1'MEM2)MPCL
      NCCR(5)=1
      MPD=MPCH*256+MPCL
      GO TO 5
612   WRITE(1,614)
614   FORMAT('INTERRUPCION(SI/NO)?',/,'..')
      READ(6,65,END=87)KL3,KL4
65    FORMAT(2A1)
      IF(KL3.EQ.LS.AND.KL4.EQ.LI)GO TO 66
      IF(KL3.EQ.LN.AND.KL4.EQ.LO)GO TO 612
      GO TO 612
66    WRITE(1,615)
615   FORMAT('CUAL INTERRUPCION?',/,'XX')
      READ(6,616,END=87)KL1,KL2
616   FORMAT(2A1)
      IF(KL1.EQ.LI.AND.KL2.EQ.LR)GO TO 617
      IF(KL1.EQ.LI.AND.KL2.EQ.LS)GO TO 618
      WRITE(1,619)
619   FORMAT('*13*ERROR EN TIPO DE INTERRUPCION')
      GO TO 612
617   IF(NCCR(5).EQ.1)GO TO 612
      MEM1=65532
      MEM2=65533
      GO TO 613
618   MEM1=65528
      MEM2=65529
      GO TO 613
62    GO TO (621,622,623,624,625,626),NREG
621   NCCR(1)=0
```

```
622     NCCR(5)=0
        GO TO 5
623     NCCR(2)=0
        GO TO 5
624     NCCR(1)=1
        GO TO 5
625     NCCR(5)=1
        GO TO 5
626     NCCR(2)=1
        GO TO 5
63      MCCRD=NAD
        GO TO 6
64      NAD=MCCRD
        GO TO 6
87      RETURN
        END
CART ID 4EE2    DB ADDR  51F0    DB CNT    0540


00/00/70  0454 HRS
PAGE 001


==>
```

```
T WRITO
WRITO.S(4EE2)
«TWO WORD INTEGERS

C      SUBRUTINA QUE ESCRIBE EN LA MEMORIA 0000

       SUBROUTINE WRITO(MPD,MOD)
       IF(MPD.NE.0)GO TO 5
       WRITE(1'65536)MOD
       GO TO 6
5      WRITE(1'MPD)MOD
6      RETURN
       END
CART ID 4EE2   DB ADDR  42B0   DB CNT   0016


00/00/70  0315 HRS
PAGE 001


==>
```

```
T ENRE
ENRE.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       SUBRUTINA QUE PONE VALORES EN REGISTROS

        SUBROUTINE ENRE(LL,NERR)
        DIMENSION LL(72),N11(9),KD08(16),NC(9),N1C(9)
        INTEGER BLANCO/' '/,COMA/','/
        DATA N11/'1','2','1','1','1','2','2','2','1'/
        DO 21 J=1,16
21      KD08(J)=0
        DO 20 I=1,9
20      CALL CARD(N11(I),N1C(I))
        READ(3'1)(NC(I),I=1,9)
        READ(5'3)KCOM1,KCOM2,NSEC,JBAS,M,K
        NERR=0
        IF(LL(3).NE.BLANCO)NERR=1
        NOF2=4
        NS=0
        NRD=9
        IF(JBAS.EQ.2)NRD=6
        DO 241 J=1,NRD
        NOF2=NOF2+NS
        NCAR=LL(NOF2)
        CALL RENUM(NCAR,NUM)
        N=M/N1C(NUM)
        DO 242 I=1,N
        IN=I+NOF2
242     KD08(I)=LL(IN)
        LN=N
        CALL CB16(KD08,NREP,LN,JBAS)
        NC(NUM)=NREP
        NS=N+2
        NORR=NOF2+N+1
        NORR1=NORR+1
        IF(LL(NORR).EQ.BLANCO.AND.LL(NORR1).EQ.BLANCO)GO TO 243
        IF(LL(NORR).NE.COMA)GO TO 244
241     CONTINUE
243     WRITE(3'1)(NC(I),I=1,9)
        GO TO 245
244     NERR=1
245     RETURN
        END
CART ID 4EE2    DB ADDR   55D0    DB CNT    0056


00/00/70  0315 HRS
PAGE 001


==>
```

```
T ULTID
ULTID.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       SUBRUTINA QUE IMPRIME LA DIRECCION EFECTIVA

        SUBROUTINE ULTID
        DIMENSION NLEC(9),NC16(9,16)
        READ(3'1)MDI,MOD,IEA
        NLEC(1)=IEA
        N=1
        READ(5'2)NNX1,NNX2,NSEC,JBAS,M
        CALL CO210(NLEC,JBAS,M,NC16,N)
        WRITE(1,12)
12      FORMAT(5X,'DIR-EFEC')
        IF(JBAS.EQ.16)GO TO 23
        WRITE(1,22)(NC16(1,J),J=1,M)
22      FORMAT(5X,16I2)
        GO TO 25
23      WRITE(1,24)(NC16(1,J),J=1,M)
24      FORMAT(5X,16A2)
25      RETURN
        END
CART ID 4EE2   DB ADDR  43B0   DB CNT   002C


00/00/70  0316 HRS
PAGE 001


==>
```

```
EXPLO.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       SUBRUTINA QUE EXPLORA LOS CARACTERES DE UN NOMBRE

        SUBROUTINE EXPLO(LEX,NT)
        DIMENSION LN(26),LD(10)
        DATA LD(1)-(10)/'1','2','3','4','5','6','7','8','9','0'/
        DATA LN(1)-(10)/'A','B','C','D','E','F','G','H','I','J'/
        DATA LN(11)-(20)/'K','L','M','N','O','P','Q','R','S','T'/
        DATA LN(21)-(26)/'U','V','W','X','Y','Z'/
        INTEGER BLANCO/' '/
        IF(LEX.EQ.BLANCO)GO TO 5
        DO 6 ID=1,10
        IF(LEX.EQ.LD(ID))GO TO 7
6       CONTINUE
        DO 8 IN=1,26
        IF(LEX.EQ.LN(IN))GO TO 9
8       CONTINUE
        NT=4
        GO TO 10
5       NT=3
        GO TO 10
7       NT=2
        GO TO 10
9       NT=1
10      RETURN
        END
CART ID 4EE2   DB ADDR  4AF0   DB CNT   0038


00/00/70  0316 HRS
PAGE 001


==>
```

```
T CARD
CARD.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       SUBRUTINA QUE CAMBIA UN CARACTER ALFANUMERICO A NUMERICO

        SUBROUTINE CARD(NCAR,NUM)
        DIMENSION NENT(10)
        DATA NENT/'0','1','2','3','4','5','6','7','8','9'/
        NUM=0
        DO5 I=1,10
        IF(NCAR.NE.NENT(I))GO TO 5
        NUM=I-1
        GO TO 6
5       CONTINUE
6       RETURN
        END
CART ID 4EE2    DB ADDR   42D0    DB CNT    0020


00/00/70  0316 HRS
PAGE 001


==>
```

```
T C0210
C0210.S(4EE1)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       SUBRUTINA QUE TRANSFORMA LOS REGISTROS A UNA BASE DADA

        SUBROUTINE C0210(NL,JBAS,M,NC16,N)
        DIMENSION NL(9),NC16(9,16),NCON(16)
        DO5 I=1,N
        NAUX=NL(I)
        CALL C0C4(NAUX,NCON,M,JBAS)
        DO 10 J=1,M
10      NC16(I,J)=NCON(J)
5       CONTINUE
        RETURN
        END
CART ID 4EE1    DB ADDR  58F0    DB CNT    001E


00/00/70  0316 HRS
PAGE 001


==>
```

```
* CB16
CB16.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       SUBRUTINA QUE CAMBIA UN NUMERO EN CUALQUIER BASE A DECIMAL

        SUBROUTINE CB16(KALF,NDID,M,JBAS)
        DIMENSION KD16(16),NDEX(4),KALF(16)
        NDID=0
        IF(JBAS.EQ.16)GO TO 6
        DO10 I=1,M
        CALL CARD (KALF(I),KD16(I))
10      NDID=(NDID)*JBAS+KD16(I)
        GO TO 7
6       DO5 J=1,M
5       NDEX(J)=KALF(J)
        CALL HEDE4(NDEX,NDID,M)
7       RETURN
        END
CART ID 4EE2    DB ADDR   4430    DB CNT    0024


00/00/70   0317 HRS
PAGE 001


==>
```

```
T COND4
COND4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C      SUBRUTINA QUE CAMBIA UN NUMERO DECIMAL A BINARIO EN ARREGLO

       SUBROUTINE COND4(IDD,L)
       DIMENSION L(8)
       IK=IDD
       DO5 I=1,8
       IP=IK/2
       L(I)=IK-IP*2
5      IK=IP
       RETURN
       END
CART ID 4EE2   DB ADDR  3F30   DB CNT   001A


00/00/70  0317 HRS
PAGE 001


==>
```

```
T NOMI
NOMI.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C
C       SUBRUTINA QUE OBTIENE EL CODIGO NMONICO
C
        SUBROUTINE NOMI(MODO,MNO,NMON)
        DIMENSION NMON(4),MNO(108,16),MOP(2)
        INTEGER BLANCO/' '/
        DO 5 I=2,108
        DO 10 N=5,15,2
        NN=N+1
        MOP(1)=MNO(I,N)
        MOP(2)=MNO(I,NN)
        M=2
        CALL HEDE4(MOP,NMD,M)
        IF(MODO.EQ.NMD)GO TO 15
10      CONTINUE
5       CONTINUE
        DO 20 K=1,4
20      NMON(K)=BLANCO
        RETURN
15      DO 25 J=1,4
25      NMON(J)=MNO(I,J)
        RETURN
        END
CART ID 4EE2   DB ADDR  5410   DB CNT   0032


00/00/70  0317 HRS
PAGE 001


==>
```

```
T REDOS
REDOS.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C       SUBRUTINA QUE REDUCE UN NUMERO MAYOR A 2 BYTES
C       SOLO A 2 BYTES

        SUBROUTINE REDOS(NRD)
6       IF(NRD.LT.2**16)GO TO 5
        NRD=NRD-2**16
        GO TO 6
5       RETURN
        END
CART ID 4EE2   DB ADDR  4230   DB CNT   0016


00/00/70  0317 HRS
PAGE 001


==>
```

```
T NUML4
NUML4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C       SUBRUTINA QUE CAMBIA UN NUMERO A UN ARREGLO LOGICO DE BITS

        SUBROUTINE NUML4(NND,LL8)
        LOGICAL LL8(8)
        DIMENSION N8(8)
        CALL COND4(NND,N8)
        DO10 I=1,8
        IF(N8(I).EQ.1)GO TO 11
        LL8(I)=.FALSE.
        GO TO 10
11      LL8(I)=.TRUE.
10      CONTINUE
        RETURN
        END
CART ID 4EE2    DB ADDR   3F50    DB CNT    0020


00/00/70  0318 HRS
PAGE 001


==>
;
```

```
T ME254
ME254.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C      SUBRUTINA QUE REDUCE LOS NUMEROS A MENORES DE 256

       SUBROUTINE ME254(NN)
6      IF(NN.LT.256)GO TO 5
       NN=NN-256
       GO TO 6
5      RETURN
       END
CART ID 4EE2   DB ADDR  40F0   DB CNT   0014


00/00/70  0318 HRS
PAGE 001


==>
```

```
T HAC4                          X40
HAC4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C       SUBRUTINA QUE OBTIENE EL HALF CARRY

        SUBROUTINE HAC4(NAX,NBX,NCX,NH)
        CALL NH4(NAX,N1)
        CALL NH4(NBX,N2)
        CALL NH4(NCX,N3)
        NR=N1+N2+N3
        IF(NR.GE.16)GO TO 5
        NH=0
        GO TO 6
5       NH=1
6       CONTINUE
        RETURN
        END
CART ID 4EE2    DB ADDR  3F70    DB CNT    0020


00/00/70  0318 HRS
PAGE 001


==>
```

```
CR4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C       SUBRUTINA QUE OBTIENE EL CARRY

        SUBROUTINE CR4(MCD,NC)
        IF(MCD.GE.256)GO TO 5
        NC=0
        GO TO 6
5       NC=1
6       RETURN
        END
CART ID 4EE2    DB ADDR  3F90    DB CNT    0016


00/00/70  0318 HRS
PAGE 001


==>
```

```
! BORR4
BORR4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C       SUBRUTINA QUE OBTIENE EL BORROW(LLEVAR)EN OPERACIONES
C       DE RESTA

        SUBROUTINE BORR4(NAX,KBX,KCX,NC)
        IF(NAX.GT.KBX+KCX)GO TO 5
        NC=1
        GO TO 6
5       NC=0
6       RETURN
        END
CART ID 4EE2    DB ADDR  5450    DB CNT    0018


00/00/70  0319 HRS
PAGE 001


==>
```

```
T NE4
NE4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C       SUBRUTINA QUE OBTIENE EL VALOR DE LA BANDERA N

        SUBROUTINE NE4(NRD,NN)
        IF(NRD.GE.128)GO TO 5
        NN=0
        GO TO 6
5       NN=1
6       RETURN
        END
CART ID 4EE2    DB ADDR  3FC0    DB CNT    0016


00/00/70  0319 HRS
PAGE 001


==>
```

```
T CER4
CER4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C       SUBRUTINA QUE OBTIENE EL VALOR DE LA BANDERA Z

        SUBROUTINE CER4(NRD,NZ)
        IF(NRD.EQ.0)GO TO 5
        NZ=0
        GO TO 6
5       NZ=1
6       RETURN
        END
CART ID 4EE2    DB ADDR  3FE0    DB CNT    0016


00/00/70  0319 HRS
PAGE 001


==>
```

```
I SOBR4
SOBR4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C       SUBRUTINA QUE OBTIENE EL VALOR DE LA BANDERA V

        SUBROUTINE SOBR4(NAX,NBX,NCX,NRD,NV)
        IF(NCX.NE.0)GO TO 7
        IF((NAX.LT.128).AND.(NBX.LT.128).AND.(NRD.GE.128))GO TO 5
        IF((NAX.GE.128).AND.(NBX.GE.128).AND.(NRD.LT.128))GO TO 5
        NV=0
        GO TO 6
5       NV=1
        GO TO 6
7       IF((NAX.GE.128).AND.(NBX.GE.128).AND.(NCX.GE.128).
       1AND.(NRD.LT.128))GO TO 5
        IF((NAX.LT.128).AND.(NBX.LT.128).AND.(NCX.LT.128).
       1AND.(NRD.GT.128))GO TO 5
6       RETURN
        END
CART ID 4EE2    DB ADDR   4000    DB CNT    0024


00/00/70  0319 HRS
PAGE 001


==>
```

```
T CONB4
CONB4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C       SUBRUTINA QUE CAMBIA UN ARREGLO DE BITS A UN NUMERO

        SUBROUTINE CONB4(L8,IPD)
        DIMENSION L8(8)
        IPD=0
        DO 10 I=1,8
        M=I-1
10      IPD=IPD+L8(I)*2**M
        RETURN
        END
CART ID 4EE2   DB ADDR  4040   DB CNT   0018


00/00/70  1010 HRS
PAGE 001


==>
```

```
T ARRE4
ARRE4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C      SUBRUTINA QUE TOMA EL BIT MENOS SIGNIFICATIVO
C      Y LE HACE UN DIGITO CORRESPONDIENTE

       SUBROUTINE ARRE4(NCCR,KCCRD)
       DIMENSION NCCR(8),KCCR(8)
       KCCR(1)=NCCR(1)
       DO5 I=2,8
5      KCCR(I)=0
       CALL CONB4(KCCR,KCCRD)
       RETURN
       END
CART ID 4EE2   DB ADDR  4090   DB CNT   001A


00/00/70  0327 HRS
PAGE 001


==>
```

```
7 LOGN4
LOGN4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C       SUBRUTINA QUE CAMBIA UN ARRELGLO LOGICO DE BITS
C       A UN NUMERO

        SUBROUTINE LOGN4(LL8,NND)
        DIMENSION N8(8)
        LOGICAL LL8(8),AX
        AX=.TRUE.
        DO 5 I=1,8
        IF(LL8(I).AND.AX)GO TO 6
        N8(I)=0
        GO TO 5
6       N8(I)=1
5       CONTINUE
        CALL CONB4(N8,N1D)
        NND=N1D
        RETURN
        END
CART ID 4EE2   DB ADDR  4060   DB CNT   0026


00/00/70  0327 HRS
PAGE 001


==>
```

```
T S0B4
S0B4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C       SUBRUTINA QUE OBTIENE EL VALOR DE LA BANDERA V
C       EN UN CASO ESPECIAL

        SUBROUTINE S0B4(NRD,NV)
        IF(NRD.EQ.128)GO TO 5
        NV=0
        GO TO 6
5       NV=1
6       RETURN
        END
CART ID 4EE2    DB ADDR   40B0    DB CNT    0018


00/00/70  0328 HRS
PAGE 001


==>
```

```
T CAR4
CAR4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C       SUBRUTINA QUE OETIENE EL VALOR DE LA BANDERA C(CARRY)
C       EN UN CASO ESPECIAL

        SUBROUTINE CAR4(NRD,NC)
        IF(NRD.EQ.0)GO TO 5
        NC=0
        GO TO 6
5       NC=1
6       RETURN
        END
CART ID 4EE2    DB ADDR   40D0    DB CNT    0018


00/00/70  0328 HRS
PAGE 001


==>
```

```
T R04
R04.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C      SUBRUTINA QUE HACE LA ROTACION A LA DERECHA DE UN NUMERO

       SUBROUTINE R04(NN8,NCCR,KDD,NCX)
       DIMENSION NN8(8),NCCR(8)
       NDX=NN8(1)
       KDD=KDD/2+NCX*128
       NCCR(1)=NDX
       RETURN
       END
CART ID 4EE2    DB ADDR  4110    DB CNT    0016


00/00/70  0329 HRS
PAGE 001


==>
```

```
T BALD4
BALD4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C       CALCULA EL VALOR DE LAS BANDERAS QUE CAMBIAN AL CARGAR
C       EL REGISTRO INDICE

        SUBROUTINE BALD4(NRR,NCCR,NRD)
        DIMENSION NCCR(8),NN8(8)
        CALL COND4(NRR,NN8)
        IF(NN8(8).EQ.1)GO TO 5
        NCCR(4)=0
        GO TO 6
5       NCCR(4)=1
6       CALL CER4(NRD,NZ)
        NCCR(3)=NZ
        NCCR(2)=0
        RETURN
        END
CART ID 4EE2    DB ADDR   4130    DB CNT    0022


00/00/70   0329 HRS
PAGE 001


==>
```

```
T REMOS
REMOS.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C       SUBRUTINA QUE DECREMENTA UN NUMERO DE DOS BYTES
        SUBROUTINE REMOS(MSD)
        MOS=2**16-1
        MSD=MSD+MOS
6       IF(MSD.LT.2**16)GO TO 5
        MSD=MSD-2**16
        GO TO 6
5       RETURN
        END
CART ID 4EE2    DB ADDR  4880    DB CNT    0018


00/00/70  0329 HRS
PAGE 001


==>
```

```
T GUARD
GUARD.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C       SUBRUTINA QUE GUARDA LOS REGISTROS EN EL STACK

        SUBROUTINE GUARD(NXD,MDI,MSD,MPC,NAD,NBD,MCCRD)
        MAXH=NXD/256
        MAXL=NXD-MAXH*256
        MPCH=MDI/256
        MPCL=MDI-MPCH*256
        WRITE(1'MSD)MPCL
        CALL REMOS(MSD)
        WRITE(1'MSD)MPCH
        CALL REMOS(MSD)
        WRITE(1'MSD)MAXL
        CALL REMOS(MSD)
        WRITE(1'MSD)MAXH
        CALL REMOS(MSD)
        WRITE(1'MSD)NAD
        CALL REMOS(MSD)
        WRITE(1'MSD)NBD
        CALL REMOS(MSD)
        WRITE(1'MSD)MCCRD
        CALL REMOS(MSD)
        RETURN
        END
CART ID 4EE2   DB ADDR  4250   DB CNT   0032


00/00/70  0329 HRS
PAGE 001


==>
```

```
T COC4
COC4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION

C      SUBRUTINA QUE A UN REGISTRO LE CONVIERTE A UNA BASE DADA

       SUBROUTINE COC4(NAUX,NCON,M,JBAS)

       DIMENSION NCON(16),NCO(4)
       IF(JBAS.EQ.16)GO TO 6
       IK=NAUX
       DO5 I=1,M
       IP=IK/JBAS
       LA=M-I+1
       NCON(LA)=IK-IP*JBAS
5      IK=IP
       GO TO 7
6      CALL DECE4(NAUX,NCO,M)
       DO8 J=1,M
8      NCON(J)=NCO(J)
7      RETURN
       END
CART ID 4EE2   DB ADDR  4320   DB CNT   002A


00/00/70  0330 HRS
PAGE 001


==>
```

```
T NH4
NH4.S(4EE2)
*TWO WORD INTEGERS
*EXTENDED PRECISION
C      SUBRUTINA QUE MODIFICA LOS OPERANDOS PARA CALCULAR EL HALF CARRY

       SUBROUTINE NH4(KX,KN)
       KN=KX
       DO 5 I=1,4
       J=8-I
       IF(KN.LT.2**J)GO TO 5
       KN=KN-2**J
5      CONTINUE
       RETURN
       END
CART ID 4EE2   DB ADDR  4160   DB CNT   001A


00/00/70  0330 HRS
PAGE 001


==>
```

APENDICE B : TABLAS.-ARCHIVOS

```
LECOD

WHAT FILE  2?>
C02P
J N
XXXYYY
002198
      ARCHIVO DE INSTRUCCIONES
POSIC    OP  CODIGO DIREC REGIS TIEMPO
    2   8B       11     1     1      2
    3   9B       11     2     1      3
    4   AB       11     3     1      5
    5   BB       11     4     1      4
    6   CB       11     1     2      2
    7   DB       11     2     2      3
    8   EB       11     3     2      5
    9   FB       11     4     2      4
   10   1B       11     5     3      2
   11   89       12     1     1      2
   12   99       12     2     1      3
   13   A9       12     3     1      5
   14   B9       12     4     1      4
   15   C9       12     1     2      2
   16   D9       12     2     2      3
   17   E9       12     3     2      5
   18   F9       12     4     2      4
   19   84       13     1     1      2
   20   94       13     2     1      3
   21   A4       13     3     1      5
   22   B4       13     4     1      4
   23   C4       13     1     2      2
   24   D4       13     2     2      3
   25   E4       13     3     2      5
   26   F4       13     4     2      4
   27   85       14     1     1      2
   28   95       14     2     1      3
   29   A5       14     3     1      5
   30   B5       14     4     1      4
   31   C5       14     1     2      2
   32   D5       14     2     2      3
   33   E5       14     3     2      5
   34   F5       14     4     2      4
   35   6F       15     3     3      7
   36   7F       15     4     3      6
   37   4F       15     5     1      2
   38   5F       15     5     2      2
   39   81       16     1     1      2
   40   91       16     2     1      3
   41   A1       16     3     1      5
   42   B1       16     4     1      4
   43   C1       16     1     2      2
   44   D1       16     2     2      3
   45   E1       16     3     2      5
   46   F1       16     4     2      4
   47   11       16     5     3      2
   48   63       17     3     3      7
   49   73       17     4     3      6
   50   43       17     5     1      2
   51   53       17     5     2      2
   52   60       18     3     3      7
   53   70       18     4     3      6
   54   40       18     5     1      2
   55   50       18     5     2      2
```

57	6A	21	3	3	7
58	7A	21	4	3	6
59	4A	21	5	1	2
60	5A	21	5	2	2
61	88	22	1	1	2
62	98	22	2	1	3
63	A8	22	3	1	5
64	B8	22	4	1	4
65	C8	22	1	2	2
66	D8	22	2	2	3
67	E8	22	3	2	5
68	F8	22	4	2	4
69	6C	23	3	3	7
70	7C	23	4	3	6
71	4C	23	5	1	2
72	5C	23	5	2	2
73	86	24	1	1	2
74	96	24	2	1	3
75	A6	24	3	1	5
76	B6	24	4	1	4
77	C6	24	1	2	2
78	D6	24	2	2	3
79	E6	24	3	2	5
80	F6	24	4	2	4
81	8A	25	1	1	2
82	9A	25	2	1	3
83	AA	25	3	1	5
84	BA	25	4	1	4
85	CA	25	1	2	2
86	DA	25	2	2	3
87	EA	25	3	2	5
88	FA	25	4	2	4
89	36	26	5	1	4
90	37	26	5	2	4
91	32	27	5	1	4
92	33	27	5	2	4
93	69	31	3	3	7
94	79	31	4	3	6
95	49	31	5	1	2
96	59	31	5	2	2
97	66	32	3	3	7
98	76	32	4	3	6
99	46	32	5	1	2
100	56	32	5	2	2
101	68	33	3	3	7
102	78	33	4	3	6
103	48	33	5	1	2
104	58	33	5	2	2
105	67	34	3	3	7
106	77	34	4	3	6
107	47	34	5	1	2
108	57	34	5	2	2
109	64	35	3	3	7
110	74	35	4	3	6
111	44	35	5	1	2
112	54	35	5	2	2
113	97	36	2	1	4
114	A7	36	3	1	6
115	B7	36	4	1	5
116	D7	36	2	2	4
117	E7	36	3	2	6
118	F7	36	4	2	5
119	80	28	1	1	2
120	90	28	2	1	3
121	A0	28	3	1	5

123	C0	28	1	2	2
124	D0	28	2	2	3
125	E0	28	3	2	5
126	F0	28	4	2	4
127	10	28	5	3	2
128	82	29	1	1	2
129	92	29	2	1	3
130	A2	29	3	1	5
131	B2	29	4	1	4
132	C2	29	1	2	2
133	D2	29	2	2	3
134	E2	29	3	2	5
135	F2	29	4	2	4
136	16	37	5	1	2
137	17	37	5	2	2
138	6D	38	3	3	7
139	7D	38	4	3	6
140	4D	38	5	1	2
141	5D	38	5	2	2
142	8C	41	1	3	3
143	9C	41	2	1	4
144	AC	41	3	1	6
145	BC	41	4	1	5
146	09	42	5	1	4
147	34	42	5	2	4
148	08	43	5	1	4
149	31	43	5	2	4
150	CE	44	1	3	3
151	DE	44	2	3	4
152	EE	44	3	3	6
153	FE	44	4	3	5
154	8E	44	1	4	3
155	9E	44	2	4	4
156	AE	44	3	4	6
157	BE	44	4	4	5
158	DF	45	2	1	5
159	EF	45	3	1	7
160	FF	45	4	1	6
161	9F	45	2	2	5
162	AF	45	3	2	7
163	BF	45	4	2	6
164	35	46	5	1	4
165	30	47	5	1	4
166	20	51	6	1	4
167	24	51	6	2	4
168	25	51	6	3	4
169	27	51	6	4	4
170	2C	51	6	5	4
171	2E	51	6	6	4
172	22	51	6	7	4
173	2F	51	6	8	4
174	23	51	6	9	4
175	2D	52	6	1	4
176	2B	52	6	2	4
177	26	52	6	3	4
178	28	52	6	4	4
179	29	52	6	5	4
180	2A	52	6	6	4
181	8D	53	6	1	8
182	6E	54	3	1	4
183	7E	55	4	1	3
184	AD	56	3	1	8
185	BD	56	4	1	9
186	01	57	5	1	2
187	3B	58	5	1	10

189	3F	61	5	1	12
190	3E	61	5	2	9
191	0C	62	5	1	2
192	0E	62	5	2	2
193	0A	62	5	3	2
194	0D	62	5	4	2
195	0F	62	5	5	2
196	0B	62	5	6	2
197	06	63	5	1	2
198	07	64	5	1	2

J N
XXXYYY

```
LCOM

WHAT FILE  5?>
COM
JN
XXYY
0221
      ARCHIVO DE COMANDOS
POS COM       CODIGO    ESPECIF      PROP1      PROP2
  2  BS            11         16          4          3
  3  BE            12         16          4          3
  4  SR            13 3415673 90          0          0
  5  VR            14          0          0          0
  6  VU            15          0          0          0
  7  VM            16          0          0          0
  8  CO            17          0          0          0
  9  DE            18          5          0          0
 10  CP            19          0          0          0
 11  UD            28          0          0          0
 12  EJ            21          1          0          0
 13  PM            22          0          0          0
 14  CM            23          0          0          0
 15  PR            24          0          0          0
 16  RN            21          2          0          0
 17  RS            21          3          0          0
 18  EI            21          4          0          0
 19  LC            25          0          0          0
 20  NL            26          0          0          0
 21  ER            27          0          0          0
JN
XXYY
/*

==>
```

```
LEROS

WHAT FILE  ??>
ERR
J N
XXYY
0113
```

ARCHIVO DE ERRORES

ERROR1 :ERROR DE SINTAXIS EN NOMBRE DE MACROCOMANDO,SE ACEPTA LETRAS CO
PRIMER CARACTER Y LETRAS Y NUMEROS EN LOS SIGUIENTES DEBIENDO S
EL NOMBRE DE 4 LETRAS.

ERROR2 :EN DEFINICION DE MACRO:BALANCEO DE PARENT. O MAS DE 60 CARACT.

ERROR3 :LIBRERIA DE MACRO LLENA,SE LLEGO AL LIMITE DE ALMACENAMIENTO D
MACROCOMANDOS Y EL MACRO NO FUE ALMACENADO.

ERROR4 :SE QUIERE CREAR UN MACROCOMANDO YA EXISTENTE EN LIBRERIA DE MAC
SI SE DESEA SOLO EJECUTAR EL MACRO PONGASE SOLO EL NOMBRE

ERROR5 :SE HA DADO COMANDO DE BORRAR UN MACROCOMANDO QUE NO EXISTE EN
LIBRERIA,O NO ESTA BIEN DEFINIDO EL COMANDO BM.

ERROR6 :EL NOMBRE DE UN MACROCOMANDO QUE SE PIDE BORRAR ES MAS DE 4 LE
O TIENE CARACTERES INVALIDOS,O EL NOMBRE EMPIEZA CON UN CARACTE
DISTINTO A UNA LETRA,O NO HAY (COMA AL INICIO)

ERROR7 :UN COMANDO NO DEFINIDO O MAL DEFINIDO EN EL SIMULADOR

ERROR8 :EL COMANDO QUE ESTA DENTRO DEL MACROCOMANDO NO HA SIDO DEFINIDO

ERROR9 :UNA DIRECCION ILEGAL DE MEMORIA FUE ENCONTRADA.LA DIRECCION NO
DENTRO DEL RANGO DE MEMORIA DEL MICROPROCESADOR (EN HEXADECIMAL
DESDE 0000 HASTA FFFF.

ERROR10:EL VALOR DE ENTRADA SI ES A MEMORIA HA SOBREPASADO 1 BYTE(FF EN
HEXADECIMAL) Y SI ES A UN REGISTRO DEPENDE DE ESTO:SI ES ACUMUL
A POR EJEMPLO NO DEBE SOBREPASAR FF EN HEXADECIMAL,O SI ES X PO
EJEMPLO NO DEBE SOBREPASAR FFFF EN HEXADECIMAL.

ERROR11:NO SE SIMULA EL PROGRAMA QUE LUEGO DE SER ENSAMBLADO TIENE ERRO

ERROR12:ERROR EN INSTRUCCION.EL SIMULADOR ENCONTRO UNO DE LOS 59 CODIG
NO DEFINIDOS,ES DECIR QUE NO CONSTITUYE UNA INSTRUCCION VALI
DEL MICROPROCESADOR

ERROR13 :ERROR EN COMANDO DE INTERRUPCION SIMULADA,UTILIZAR IR:PARA
INTERRUPCION DE HARDWARE E IS PARA INTERRUPCION SIN MASCARA.

J N
%XYY
/*

==>

APENDICE C

MANUAL DE USUARIO

El usuario dispone para el manejo del Simulador del M6800 de comandos, macrocomandos y comandos a la tabla de macros.

Para ingresar al Simulador, se pone el nombre del programa y los nombres de los archivos utilizados así:

S6800

FILE 1 : MEMRO

FILE 2 : CO2P

FILE 3 : REGIS

FILE 4 : (Nombre del programa ensamblado)

FILE 5 : COM

FILE 6 : MACRO

FILE 7 : ERR

FILE 8 : NMONI

COMANDOS DISPONIBLES : Se indicará aquí los formatos de los comandos unicamente, ya que su función fue indicada en 3-4

-COMANDOS PARA CONTROL DE FORMATOS E IMPRESION DE SALIDAS

BE :Base de Entrada

Formato : BE02 Base de entrada en binario

BE08 Base de entrada en octal

BE16 Base de entrada en decimal

BE10 Base de entrada en decimal

BS :Base de Salida

Formato : Tiene el mismo formato que BE

SR :Selección de Registros

Formato : SR IOEPXABCS

Aquí, I : Dirección de la Instrucción

E : Dirección efectiva del operando

O : Operador

P : Contador del Programa

X : Registro Indice

A : Acumulador A

B :Acumulador B

C : Registro de Condiciones

S : Registro puntero del 'stack'

DE : Da encabezamiento de nombres de registros

Formato : DE NUM

Función : Saca el rotulado de registros cada NUM líneas

LC : Lista comandos cada vez que se vaya a ejecutar

Formato : LC

NL : No lista comandos

Formato : NL

CO : Código Objeto

Formato : CO OO Escoge Código Objeto en impresión de instrucciones que se ejecuten.

CO O1 Escoge Código Nmónico

-COMANDOS PARA PONER VALORES

PM :Poner en Memoria

Formato : PM LOC,PL1,PL2,PL3,....

El formato esta de acuerdo a la base de entrada :

BE02 PM 0000000010101010,00000001,...(4 valores)

BE08 PM 000777,123,456,.....(10 valores)

BE16 PM 0A00,AA,BB,0C,0D,0F,.....(18 valores)

BE10 PM 000128,016,012,.....(12 valores)

Función : La localidad LOC es puesta al valor PL1 y las siguientes localidades a PL2,PL3,etc.

PR : Poner en Registros

Formato : PR RVALOR1,RVALOR2

El formato es de acuerdo a la base de entrada :

BE16 PR I0508,A06,B1C,C1E,.....(9 valores)

BE08 PR I121716,B121,C128,.....(9 valores)

BE02 PR A01010101,B11111111,.....(4 valores)

BE10 PR A123,B234,C255,.....(9 valores)

CM : Continúe poniendo en memoria

Formato : CM PL1,PL2,...

El formato está de acuerdo a la base de entrada.

Función : Pone el valor PL1 a partir de la última localida de memoria ingresada en el comando PM;PL2 en la siguiente y así sucesivamente.

CP : Cargar el programa ensamblado

Formato : CP

-COMANDOS PARA VISUALIZACION DE VALORES

VM : Ver Memoria

Formato : VM INIC,NUM

El formato es de acuerdo a la base de entrada.

BE02 VM 0101010101010101,01010101

BE08 VM 065432,123

BE16 VM 0A0A,0B

BE10 VM 032520,100

Función :Muestra NUM palabras de memoria,empezando en la localidad INIC.

VR : Ver Registros

Formato : VR

VU : Ver Ultima Instrucción ejecutada

Formato : VU

UD : Ultima Dirección de la última instrucción ejecutada

Formato : UD

ER : Mensaje de Error

Formato : ER NN Se desea mensaje del error NN

Ejemplo : ER 02 Se pide mensaje del error 2

-COMANDOS DE EJECUCION DE PROGRAMA

EJ : Ejecuta el Programa

Formato : EJ LOC,NUM (De acuerdo a la base de entrada)

Ejecuta NUM pasos de programa empezando en la localidad LOC;luego de esto imprime los registros.

Ejemplo : EJ 00AA,00FF (Ejecuta 00FF pasos desde 00AA)

RN : Rastrea N pasos del programa

Formato : RN LOC,NUM

El formato es de acuerdo a la base de entrada e igual que el comando EJ.

Función : NUM pasos de programa son rastreados a partir de la localidad LOC,siendo todos los registros impresos luego de ejecutar cada instrucción.

RS : Rastrea N saltos de programa

Formato : RS LOC,NUM

El formato es de acuerdo a la base de entrada e igual que para el comando EJ.

EI : Ejecuta Instrucciones

Formato : Igual que para el comando EJ.

-COMANDOS DE INTERRUPCION SIMULADA

IR : Interrupción de hardware simulada

Formato : IR

Se utiliza luego de una instrucción WAI.

IS : Interrupción sin máscara

Formato : IS

Se utiliza luego de una instrucción WAI ,no toma en cuenta la bandera de interrupción.

-MACROCOMANDOS

Un macrocomando es una abreviación de una serie de comandos del Simulador en un nombre definido.Para ejecutar este grupo de comandos,solamente se ingresa el nombre.

La forma de definición de macros es así :

Nombre (tira)

El nombre de un macrocomando puede contener 4 caracteres máximo,debiendo empezar con un caracter alfabético , mientras que los siguientes pueden ser números o letras. El nombre, no puede empezar con las letras de un comando, ya que este tiene prioridad sobre los macrocomandos.

La tira es una secuencia de comandos que van separados por puntos,sin permitir espacios en blanco luego o antes de los puntos.

La tira de comandos va encerrada entre paréntesis, y va separada del nombre del macrocomando por un espacio en blanco.

El macrocomando puede tener una extensión máximá de 60 caracteres.

El límite de macrocomandos a guardarse es de 29; ocurriendo un error ,si se desea crear un macrocomando luego de que esté llena la tabla de macros.

Ejemplo : Sea el nombre de un macrocomando :GAMA , que realize las siguientes funciones :

-Base de Entrada 16

-Poner en Registros A y B el valor FF

-Ver Registros

La definición del macrocomando se hará así:

GAMA (BE16.PR AFF,BFF.VR)

La tira de comandos se ejecuta poniendo el nombre:

GAMA

-COMANDOS A LA TABLA DE MACROS

LM : Liste Macros

Formato : LM

BM : Borre Macros

Formato : BM,NAM1,NAM2,....(9 nombres)

Borra los macros NAM1,NAM2,etc.

Para salirse del Simulador se ingresa en vez de un comando,los símbolos /X.

B I B L I O G R A F I A

-GORDON GEOFFREY, Simulación de Sistemas, Diana, México, 1981.

-NAYLOR THOMAS, Técnicas de Simulación en Computadoras, Limusa, México, 1980.

-GRIES D. , Construcción de compiladores, Paraninfo, España, 1974.

-MOTOROLA SEMICONDUCTOR PRODUCTS INC, Microprocessor Applications Manual, Mc Graw Hill, 1975.

-SHERMAN, Técnicas de Programación en computadoras , PHI, España, 1973.

-DONOVAN, Programación de Sistemas, El Ateneo, Argentina, 1981.

-HERNANDEZ M, Traductor Cross-Assembler para el Microprocesador M6800 (Tesis de Ing. Eléctrica).

-CYTOS DM-250, Reference Manual, 1978.

-DNA FORTRAN IV, Reference Manual, 1978.